TÍTULO: Sistema de autocalibración de cámaras y reconstrucción 3D

AUTOR: Guillermo Gallego Bonet

TUTOR: José Ignacio Ronda Prieto

DEPARTAMENTO: Señales, Sistemas y Radiocomunicaciones

TRIBUNAL:

Presidente	D. Fernando Jaureguizar Núñez
Vocal	D. José Ignacio Ronda Prieto
Vocal Secretario	D. Francisco Morán Burgos
Suplente	D. Luis Salgado Álvarez de Sotomayor

FECHA DE LECTURA: 23 de febrero de 2004

CALIFICACIÓN: Matrícula de Honor, 10p.

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

SISTEMA DE AUTOCALIBRACIÓN DE CÁMARAS Y RECONSTRUCCIÓN 3D

GUILLERMO GALLEGO BONET

Febrero de 2004

RESUMEN DEL PROYECTO:

El presente Proyecto Fin de Carrera desarrolla un sistema completo para la autocalibración de cámaras y la reconstrucción espacial de escenas formadas por puntos 3D. La entrada del sistema consiste en ficheros con especificaciones de puntos 2D previamente localizados en imágenes, sin que esta localización forme parte del proyecto. La salida consiste en los parámetros intrínsecos y extrínsecos de las cámaras y la escena 3D. Todo el sistema se implementa en el entorno de programación conocido como MATLAB.

El sistema implementa las principales alternativas para los algoritmos de autocalibración y reconstrucción conocidas en la literatura. También se realiza una evaluación experimental cuyo objetivo es la comparación de la eficacia de las distintas técnicas utilizadas.

PALABRAS CLAVE:

Imagen, cámara, autocalibración, reconstrucción tridimensional, geometría proyectiva, optimización, parámetros intrínsecos y extrínsecos, matriz fundamental, tensor trifocal, tensor cuadrifocal, ajuste de haces, horóptera.

Agradecimientos

Este proyecto constituye el fin de una etapa y el comienzo de otra. La etapa que toca a su fin es el resultado de la historia particular de la vida del autor. A lo largo de esta vida son muchas las personas que han influido en su forma de ser, especialmente sus educadores. A pesar de que el proyecto es un documento de carácter técnico, permítanme utilizar unas pocas líneas para expresar mi agradecimiento hacia las personas que lo han hecho posible. Es el momento de poner el toque emotivo al proyecto.

Así que quisiera comenzar esta sección recordando a mis padres, Guillermo y Mª del Carmen: gracias por vuestro cariño, por la educación que me habéis ofrecido, por facilitarme tanto la vida y por los genes (me gusta mi forma de ser). Gracias a mis hermanas, Amparo y Mª del Carmen (Mela), por formar una *familia*: sin vosotras no soy nadie. Gracias a la familia con la que más convivo: mi padrino José Luis, la tía Inma (la madrina), y mis primos José Luis, Inmaculada y Ramón, junto con la reciente incorporación de Alfonso, porque nos habéis enseñado que no somos tres, sino seis. Gracias a mi madrina, Paloma, y el resto de tíos, primos y vecinos, por preocuparos por mí y crear un ambiente propicio para mi aprendizaje.

Gracias a mis amigos de la escuela, que me habéis ayudado a superar esta difícil prueba durante el turno de tarde: Fernando, Daniel, David, Enrique, Jaime, Ángel, Hugo, Eloy, Carlos, Roberto, Pedro, etc. Son muchos los días que hemos pasado juntos y las veces que hemos padecido por los exámenes. Tampoco me olvido de los que me habéis acompañado no tan de cerca.

No se entendería el tema de este proyecto sin conocer a mi tutor, José Ignacio Ronda. Gracias por hacer el esfuerzo de cambiar del campo de la codificación de vídeo al de la visión por ordenador. Gracias por todas las horas dedicadas, por tus enseñanzas, por poner a mi alcance todos los medios y tecnología adecuada, por tus ánimos y por la confianza depositada en mí desde que nos conocimos en tercero de carrera.

También quisiera expresar mi gratitud hacia el resto de miembros del Grupo de Tratamiento de Imágenes, que me habéis acogido y escuchado siempre que lo he necesitado: Narciso, Fernando, Francisco, Luis, Julián, José Manuel, Enrique, etc. junto el resto de personal: Patricia, María José, Rebeca y mis compañeros de laboratorio: Ainhoa, Enrique, José, José María, Juan, Juan José, Pablo, Silvia, etc.

Gracias también a Antonio Valdés y Jesús Ruiz, profesores de la facultad de Matemáticas de la UCM, por sus clases de geometría proyectiva y diferencial. Porque en verdad este proyecto tiene mucho más contenido geométrico que análisis armónico, a pesar de que me gusten ambos campos y haya cursado con agrado muchas más asignaturas sobre el segundo tema.

También quiero recordar a mis amigos del Coro Guadalupe, del cual formo parte y a mi profesor de guitarra, Ricardo García, porque no todo en la vida es estudiar la carrera: también hay que invertir tiempo en otras cosas, como la música.

Gracias a mis compañeros y profesores del colegio Sagrado Corazón de Jesús, que hicieron un buen trabajo conmigo, especialmente mis profesores de Judo: Rafael Ortega y Purificación Polo, que desde los 6 hasta los 19 años me habéis enseñado la disciplina y el resto de valores de ese deporte: la cortesía, el coraje, la modestia, el respeto, el control de sí, la amistad, el honor y la sinceridad.

Y sobre todo, gracias a ti, Esther, por estar siempre a mi lado, por la paciencia e interés con que me escuchas, por iluminar mi vida desde que nos conocimos en primero de carrera.

Este proyecto ha sido subvencionado mediante una beca del proyecto de investigación TIC2001−3069.

Hay una fuerza motriz más poderosa que el vapor,
la electricidad y la energía atómica: la voluntad.

Albert Einstein.

Este proyecto está dedicado a mi familia.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

El presente proyecto está englobado dentro del campo de la visión artificial, la cual pretende emular el comportamiento de la visión humana utilizando una cámara como sensor y un ordenador como procesador. El proyecto tiene como objetivo la autocalibración de un conjunto de cámaras y la reconstrucción 3D de la estructura de la escena adquirida.

Suponemos que se dispone de un conjunto de imágenes digitales o una secuencia de vídeo que refleja una escena estática. A partir de dichas imágenes se extraen características (puntos y/o rectas) y se pretende obtener un modelo en tres dimensiones de la escena (principalmente basado en la localización en el espacio de ciertos puntos significativos), suponiendo que se desconocen las características de la cámara con que fueron adquiridas las imágenes, así como la posición espacial de la misma respecto de la escena durante la adquisición: no hay ningún tipo de conocimiento *a priori*, sólo la información contenida en las imágenes.

Los dos elementos principales del título del proyecto ya han sido presentados en el párrafo anterior: el término *autocalibración* significa obtener los parámetros que definen la cámara (distancia focal, punto principal, posición del espacio, etc.) sin ningún conocimiento *a priori* de la escena adquirida, mientras que el término *reconstrucción 3D* significa obtener un modelo tridimensional de la escena bajo estudio: forma y localización de los objetos que están delante de la cámara.

No es objeto del proyecto la extracción de características de las imágenes digitales de partida, ni su correspondencia entre imágenes, ni la representación tridimensional mediante mallado y pegado de texturas sobre la superficie reconstruida. Así pues, el objetivo es abordar las etapas intermedias entre estos extremos.

La memoria está organizada de la siguiente manera: en el capítulo 2 se describe el problema que se pretende resolver y se exponen el esquema de procesamiento y el modelo físico utilizado; en el capítulo 3 se realiza una introducción a la geometría proyectiva, herramienta matemática con la que se describe la mayor parte del modelo físico. En el capítulo 4 se trata un tema básico para desarrollar el resto de elementos geométricos como es la estimación de homografías en el plano. El capítulo 5 toca superficialmente los elementos de la matriz de proyección que representa una cámara y se aprende a estimarla.

La calibración proyectiva es la idea central del capítulo 6, en el que se tratan temas como la matriz fundamental, la triangulación, el tensor trifocal, el tensor cuadrifocal, la geometría multicámara, el ajuste de haces y la distorsión radial. El capítulo 7 versa sobre la autocalibración de cámaras, en él se introducen algunos algoritmos para estimar los parámetros intrínsecos y extrínsecos de las cámaras. El capítulo 8 trata sobre la optimización posterior a la calibración euclídea.

Al final de los capítulos más importantes se exponen algunos resultados experimentales de los algoritmos descritos, de los que se pueden extraer las principales conclusiones de los temas tratados.

En el último capítulo se procede a una breve discusión de los resultados y las líneas de trabajo futuro en respuesta a las debilidades encontradas. Los apéndices incluidos al final de la memoria tratan temas secundarios que no queremos que distraigan la atención del lector respecto del argumento principal. En ellos se exponen: la notación seguida, algoritmos de optimización, descripción de las rutinas implementadas, etc.

Antes de entrar en detalles sobre cada uno de los temas, me gustaría citar algunas de las aportaciones innovadoras que el autor ha desarrollado en el proyecto: una modificación del algoritmo lineal de cálculo de homografías en el plano § 4.2, un algoritmo robusto de calibración proyectiva § 6.6.4.1, un nuevo algoritmo de autocalibración para cámaras con parámetros intrínsecos constantes basado en la estimación de la cuádrica absoluta mediante horópteras § 7.8.6, un ajuste de haces proyectivo que compensa la distorsión radial § 6.6.5 y un ajuste de haces euclídeo adaptado a cámaras con píxeles de forma conocida § 8.2. Además, se ha hecho el esfuerzo de expresar la mayoría de los problemas de optimización en un marco descriptivo común § 4.3, por lo que a lo largo de la memoria se reformulan muchos algoritmos en términos de una *función modelo*, en lugar de una *función de coste*.

En verdad, el algoritmo de autocalibración de § 7.6.2 no está recogido en ningún artículo, pero es una pequeña variación sobre otro existente y no es la primera vez que uno se da cuenta de que ha reinventado la rueda. El análisis experimental *completo* presentado desde un *enfoque común* que se hace en el capítulo de la calibración proyectiva § 6.7 no tiene precedente en la literatura.

Capítulo 2

Planteamiento del problema y modelo utilizado

2.1. El problema de la reconstrucción 3D

El problema que consideramos consiste en obtener a partir de un conjunto de imágenes

- la estructura tridimensional de la escena captada,
- los parámetros externos de la cámara que definen su movimiento,
- y los parámetros internos de la misma.

En algunas ocasiones supondremos conocidos algunos de los valores de los parámetros de las cámaras. En adelante, suponemos que ya han sido detectadas ciertas características entre imágenes y puestas en correspondencia, a través de algún algoritmo de seguimiento. A partir de este número finito de características pretendemos resolver nuestro problema, además de imponer ciertas restricciones, si no sería imposible reconstruir nuestro mundo físico, ya que su complejidad es infinitamente superior a la complejidad de las medidas que podamos hacer en las imágenes.

Así pues, queda establecido el dominio del proyecto: la entrada son un conjunto de correspondencias de características entre imágenes (puntos y/o rectas) y la salida son los parámetros intrínsecos y extrínsecos de la cámara, junto con las posiciones de los puntos 3D.

Habitualmente, la resolución de este problema se divide en varias etapas y los posibles resultados intermedios de cada una de ellas son lo que se conoce como una reconstrucción proyectiva o una reconstrucción afín de la escena. Estas fases reciben el nombre de reconstrucción o calibración *estratificada*. Ambas reconstrucciones están relacionados con la reconstrucción final, denominada reconstrucción euclídea o métrica, mediante una transformación geométrica del espacio: proyectiva o afín, respectivamente. Las transformaciones proyectivas también reciben el nombre de homografías, término que será explicado más adelante.

Comenzamos estableciendo el modelo del proceso físico de captación de imágenes y la terminología básica del problema.

2.2. Modelo y definiciones básicas

En este apartado queremos ver la relación que hay entre el mundo real y las imágenes que podemos tomar con una cámara.

Todos somos conscientes de que perdemos una dimensión al proyectar el mundo real sobre una imagen. El concepto de "perder una dimensión" tiene la siguiente explicación matemática: el mundo real es comúnmente identificado con $\mathbb{R}^3$ gracias a la fiel descripción local que proporciona (en ausencia de tiempo), mientras que las imágenes son planas, son un pequeño trozo del plano $\mathbb{R}^2$. Esta caída de superíndice: del 3 al 2 es lo que entendemos como "perder una dimensión". Nuestra misión es tratar de recuperar esta dimensión perdida, inferirla a partir de varias proyecciones, tomadas desde distintas posiciones del espacio.

Una cámara es un dispositivo que realiza esta proyección del mundo tridimensional al mundo de las imágenes, de dominio bidimensional, y su acción recibe el nombre de proceso de formación de la imagen. Este proceso es complejo porque intervienen muchos elementos: un conjunto de lentes se utiliza para dirigir la luz, controlar su dirección de propagación, para que incida sobre un dispositivo CCD y convierta la información en bits. Los fundamentos ópticos de la formación de la imagen habitualmente suponen un modelo de *lente delgada* o fina, lo que simplifica el análisis: despreciamos los efectos de difracción y reflexión de las lentes, y nos quedamos con el modelo más sencillo de refracción.

Más aún, si hacemos que la apertura de la lente disminuya a cero, esto fuerza que todos los rayos pasen por el centro óptico de la lente, conocido como modelo ideal de la cámara de ojo de aguja (*pinhole*). Sin embargo hay un problema: a medida que la apertura disminuye la difracción se vuelve dominante, por consiguiente, el modelo de lente delgada (puramente refractivo) no es aplicable. Además, a medida que la apertura disminuye, también lo hace la energía que atraviesa la lente. Aunque actualmente es posible construir dispositivos que se aproximan a cámaras de ojo de aguja, desde nuestro punto de vista el modelo de ojo de aguja constituirá una buena aproximación geométrica de un sistema de adquisición de imágenes bien enfocado.

El modelo matemático más popular que describe esta operación es el de cámara proyectiva, que corresponde a una proyección cónica, por lo que sus elementos básicos son el *plano de proyección*, *plano de la imagen* o retina y el *centro de proyección*.

En primer lugar queremos obtener las ecuaciones de la proyección en sistemas de referencia adaptados que hagan estas ecuaciones lo más sencillas posible. Tomemos como referencia espacial un sistema ortonormal con origen en el centro óptico de la proyección, cuyo eje z esté alineado con el eje óptico de la lente y los otros dos ejes sean paralelos al plano de proyección. Sea la referencia en el plano un sistema con origen en el punto principal y ejes paralelos a los del sistema espacial. El punto principal es el punto de intersección del eje óptico con el plano de la imagen. La figura 2.1 representa la notación de ejes utilizada.

Un punto p del espacio, de coordenadas $\widetilde{\mathbf{X}} = (X, Y, Z)^\top$ y su proyección en el plano $\widetilde{\mathbf{x}} = (x, y)^\top$ guardan la siguiente relación, deducida por semejanza de triángulos,

$$x = -f\frac{X}{Z}, \qquad y = -f\frac{Y}{Z} \tag{2.1}$$

donde f es la *distancia focal*. A veces se utiliza la notación y símbolo de aplicación proyección π:

$$\begin{array}{rccc} \pi: & \mathbb{R}^3 \backslash \{Z=0\} & \longrightarrow & \mathbb{R}^2 \\ & \widetilde{\mathbf{X}} & \mapsto & \widetilde{\mathbf{x}} = \pi(\mathbf{X}) \end{array}$$

Cualquier otro punto sobre la recta que pasa por el centro óptico y por p se proyecta sobre las mismas coordenadas $\mathbf{x}$, esto refleja el modelo de la cámara de ojo de aguja.

Obsérvese que hay un signo negativo en cada igualdad de (2.1). Esto hace que la imagen de un objeto esté invertida (boca abajo) en el plano de la imagen. Para eliminar este efecto, cambiamos de signo las coordenadas, que es equivalente a suponer el plano de la imagen está en $z = +f$, en lugar de en $z = -f$. A veces se denomina a este último "plano virtual de la imagen", ya que está situado delante

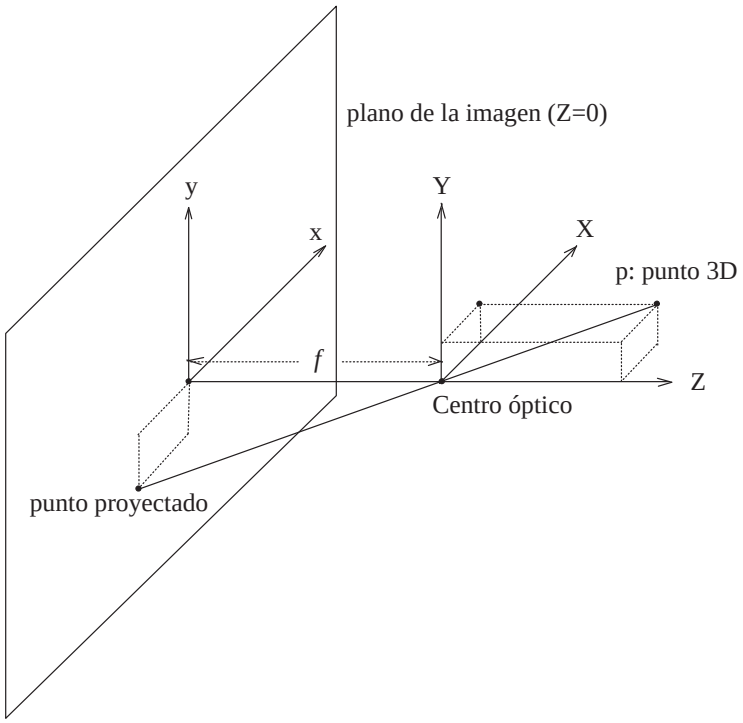


Figura 2.1: Sistemas de coordenadas adaptados a la cámara

de la lente. En adelante utilizaremos este modelo, con el plano de la imagen delante del centro óptico, ya que resulta más cómodo. Las nuevas coordenadas de la proyección del punto p son:

$$x = f\frac{X}{Z}, \qquad y = f\frac{Y}{Z} \tag{2.2}$$

Son necesarios varios cambios de referencia, tanto en el espacio como en el plano para establecer una correspondencia precisa entre puntos del espacio 3D, expresados respecto de un sistema de referencia global fijo, y sus proyecciones en la imagen, expresadas respecto de un sistema de referencia local a la imagen. Existen tres tipos de transformaciones involucradas:

- Cambio de base en el espacio entre el sistema de referencia de la cámara y el de los objetos.
- Proyección de coordenadas espaciales a coordenadas en un plano.
- Cambio de base en el plano de la imagen entre distintos sistemas de referencia.

A continuación veremos que podemos expresar este proceso simplificado de formación de la imagen y cambios de coordenadas mediante la multiplicación encadenada de varias matrices, por lo que comenzaremos a utilizar el álgebra matricial para desarrollar el modelo matemático. Invertir esa cadena de transformaciones se conoce como "calibración de la cámara", que es un paso clave hacia la reconstrucción 3D.

A pesar de que las ecuaciones (2.2) son intrínsecamente no lineales, al pasar a coordenadas homogéneas, las ecuaciones se vuelven lineales. Las coordenadas homogéneas son propias de los espacios proyectivos. Un punto del espacio $\mathbb{R}^3$, de coordenadas $\widetilde{\mathbf{X}} = (X, Y, Z)^\top$, se representa en coordenadas homogéneas añadiendo un 1 como última coordenada: $\mathbf{X} = (X, Y, Z, 1)^\top$, obteniendo así un vector de 4 componentes. Además, se establece que dos vectores en coordenadas homogéneas representan el mismo punto si son proporcionales. A simple vista este cambio puede parecer perjudicial, ya que se añade

una coordenada más, sin embargo comprobaremos que es muy ventajoso.

Si expresamos las ecuaciones de la proyección cónica en coordenadas homogéneas, éstas se pueden expresar de forma lineal:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

El símbolo $\sim$ significa igualdad *salvo proporcionalidad*. En la anterior ecuación podemos descomponer la matriz:

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \mathtt{K}_f[\mathtt{I} \mid \mathbf{0}]$$

donde $\mathtt{K}_f$ es la matriz diagonal $\mathrm{diag}(f, f, 1)$ y $[\mathtt{I} \mid \mathbf{0}]$ es la matriz de proyección canónica. De forma compacta la ecuación de la proyección es:

$$\mathbf{x} \sim \mathtt{P}\mathbf{X}$$

siendo $\mathbf{x} = (x, y, 1)^\top$ las coordenadas homogéneas del punto proyectado en la imagen y $\mathtt{P} = \mathtt{K}_f[\mathtt{I} \mid \mathbf{0}]$ la matriz de la cámara proyectiva.

2.2.1. Parámetros intrínsecos

Consideramos dentro de la imagen dos sistemas de referencia afines:

- Un sistema arbitrario en el que vienen expresados los datos. Notemos mediante (u, v) las coordenadas de un punto en este sistema, a las que llamaremos coordenadas de píxel.
- Un sistema ortonormal con origen en el punto principal y uno de sus vectores paralelo a uno del sistema anterior, con coordenadas asociadas (x, y), llamadas coordenadas normalizadas.

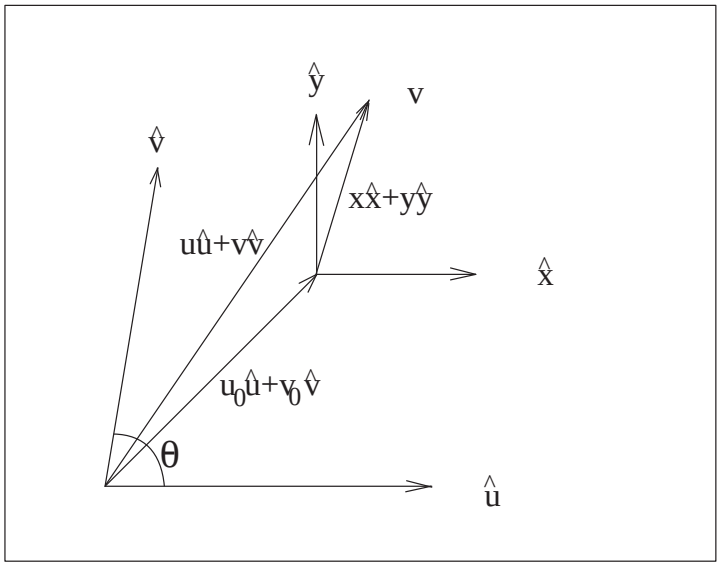


Figura 2.2: Sistemas de coordenadas en el plano

En la figura 2.2 están representados ambos sistemas. Ahora hallaremos la matriz que expresa la relación entre las coordenadas de ambos sistemas. Para ello utilizamos la relación vectorial presentada en la misma figura.

$$u_0\vec{u} + v_0\vec{v} + x\vec{x} + y\vec{y} = u\vec{u} + v\vec{v} \tag{2.3}$$

llamemos m_u y m_v a las dimensiones de los vectores $\vec{u}$ y $\vec{v}$ en unidades de los vectores $\vec{x}$ e $\vec{y}$ respectivamente, y sea θ el ángulo que forman, resulta la siguiente relación:

$$\begin{aligned} \vec{u} &= m_u\vec{x}; \\ \vec{v} &= m_v\cos\theta\vec{x} + m_v\,\mathrm{sen}\,\theta\vec{y}; \end{aligned}$$

y sustituyendo en (2.3), se deduce

$$\begin{aligned} x &= (u-u_0)m_u + (v-v_0)m_v\cos\theta; \\ y &= (v-v_0)m_v\,\mathrm{sen}\,\theta; \end{aligned}$$

Despejemos u y v

$$\begin{aligned} u &= u_0 + m_u^{-1}x - m_u^{-1}(\tan\theta)^{-1}y \\ v &= v_0 + m_v^{-1}(\mathrm{sen}\,\theta)^{-1}y \end{aligned}$$

matricialmente, vemos que sigue el patrón de una transformación afín: $\widetilde{\mathbf{u}} = \mathtt{A}\widetilde{\mathbf{x}} + \widetilde{\mathbf{b}}$,

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} m_u^{-1} & -m_u^{-1}(\tan\theta)^{-1} \\ 0 & m_v^{-1}(\mathrm{sen}\,\theta)^{-1} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}.$$

Esto mismo, expresado en coordenadas homogéneas:

$$\mathbf{u} = \begin{bmatrix} \mathtt{A} & \widetilde{\mathbf{b}} \\ 0 & 1 \end{bmatrix}\mathbf{x}$$

es decir,

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} m_u^{-1} & -m_u^{-1}(\tan\theta)^{-1} & u_0 \\ 0 & m_v^{-1}(\mathrm{sen}\,\theta)^{-1} & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

Incluyendo esto en las ecuaciones de la proyección cónica que ya conocemos

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{bmatrix} m_u^{-1} & -m_u^{-1}(\tan\theta)^{-1} & u_0 \\ 0 & m_v^{-1}(\mathrm{sen}\,\theta)^{-1} & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} \frac{f}{m_u} & -\frac{f}{m_u}\cot\theta & u_0 & 0 \\ 0 & \frac{f}{m_v\,\mathrm{sen}\,\theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Denominando $\mathtt{K}$ a la matriz cuadrada formada por las tres primeras columnas de la última matriz, y notando $\mathbf{u} = (u, v, w)^\top$, la ecuación queda de forma concisa:

$$\mathbf{u} \sim \mathtt{K}[\mathtt{I} \mid \mathbf{0}]\mathbf{X}_{\mathrm{cam}}$$

Denotamos $\mathbf{X}_{\mathrm{cam}} = (X, Y, Z, 1)^\top$ para enfatizar que, por ahora, la referencia espacial utilizada está adaptada a la cámara.

La matriz $\mathtt{K}$ se denomina *matriz de parámetros intrínsecos* de la cámara y depende de cinco parámetros: $\alpha_u = f/m_u$, la relación de aspecto $\tau = m_v/m_u$, el ángulo θ (que define la inclinación entre los lados de los píxeles - *skew*), y las coordenadas del punto principal (u_0, v_0). Si llamamos $\alpha_v = f/m_v$, la relación de aspecto también se expresa como $\tau = \alpha_u/\alpha_v$.

$$\mathtt{K} = \begin{bmatrix} \alpha_u & -\alpha_u\cot\theta & u_0 \\ 0 & \alpha_v/\,\mathrm{sen}\,\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \tau\alpha_v & -\tau\alpha_v\cot\theta & u_0 \\ 0 & \alpha_v/\,\mathrm{sen}\,\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

2.2.2. Parámetros extrínsecos

En general, los puntos del espacio están expresados respecto de un sistema de referencia ortonormal arbitrario distinto del que está adaptado a la cámara, al que se denomina: sistema de referencia del mundo. Ambos sistemas de coordenadas espaciales se relacionan mediante una transformación propia de un mundo euclídeo en el que sólo existen movimientos rígidos : una transformación euclídea, es decir, una rotación y una traslación.

Si $\widetilde{\mathbf{X}}$ son las coordenadas de un punto en el sistema de referencia del mundo y $\widetilde{\mathbf{X}}_{\text{cam}}$ son las coordenadas del mismo punto, pero en el sistema de referencia adaptado a la cámara, entonces podemos expresar la relación entre ellos: $\widetilde{\mathbf{X}}_{\text{cam}} = \mathtt{R}(\widetilde{\mathbf{X}} - \widetilde{\mathbf{C}})$, donde $\widetilde{\mathbf{C}}$ son las coordenadas del centro óptico de la cámara en el sistema de referencia del mundo y $\mathtt{R}$ es una matriz de rotación que representa la orientación del sistema de referencia adaptado a la cámara respecto del sistema de referencia del mundo. Esta relación en coordenadas homogéneas es:

$$\mathbf{X}_{\text{cam}} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} \mathtt{R} & -\mathtt{R}\widetilde{\mathbf{C}} \\ \mathbf{0}^\top & 1 \end{bmatrix} \mathbf{X} = \mathtt{H}_e \mathbf{X}$$

La matriz de la transformación euclídea, $\mathtt{H}_e$, define el movimiento rígido entre ambos sistemas de referencia. La traslación la denotamos de forma general mediante el vector $\mathbf{t} \in \mathbb{R}^3$.

2.2.3. Matriz de proyección completa

El proceso de formación de la imagen completo incluye la proyección cónica sencilla y también matrices que modelan la calibración interna y externa:

$$\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]\mathtt{H}_e = \begin{bmatrix} \alpha_u & -\alpha_u \cot\theta & u_0 \\ 0 & \alpha_v / \sin\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} = \mathtt{KR}[\mathtt{I} \mid -\widetilde{\mathbf{C}}]$$

La ecuación de la proyección respecto de los nuevos sistemas de referencia de la imagen y del mundo es:

$$\mathbf{u} \sim \mathtt{KR}[\mathtt{I} \mid -\widetilde{\mathbf{C}}]\mathbf{X}$$

A veces se utiliza la letra $\mathbf{x}$ para referirse a las coordenadas homogéneas de los puntos en las imágenes respecto del nuevo sistema de referencia, en lugar de la letra $\mathbf{u}$, siguiendo la notación de [1].

2.2.4. Distorsión radial

Hasta ahora se ha supuesto que un modelo lineal es un modelo preciso del proceso de formación de la imagen. Así, los puntos 3D, punto en la imagen y centro óptico están alineados, y las rectas del mundo real se proyectan en rectas en el plano de la imagen. Para lentes reales esta suposición no se cumple. En general, la desviación más relevante es una distorsión radial. En la práctica, este error es más acusado cuanto menor es la distancia focal y peor es la calidad de la lente.

Es posible corregir la distorsión radial producida en las imágenes para seguir utilizando la suposición de cámara lineal. La distorsión debería ser corregida en el lugar correcto en la cadena de transformaciones del proceso de proyección: dicho lugar es la proyección inicial del objeto al plano de la imagen, antes de aplicar la matriz de parámetros intrínsecos. Es decir, se debería trabajar en coordenadas normalizadas. Sin embargo, en la mayor parte del proyecto se desconoce la matriz de parámetros intrínsecos de la cámara, por lo que se trabaja en coordenadas de píxel. En estas coordenadas, el modelo matemático de distorsión radial es el recogido en [1].

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + L(r) \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} \tag{2.5}$$

donde (x_d, y_d) son las coordenadas distorsionadas, observadas (medidas en la imagen), (x, y) son las coordenadas corregidas, (x_c, y_c) es el centro de distorsión radial y $r^2 = (x - x_c)^2 + (y - y_c)^2$. Si la relación de aspecto no es la unidad es necesario corregir la fórmula de cálculo de r. Las coordenadas corregidas están relacionadas con las coordenadas del punto 3D mediante una proyección lineal.

La elección de la función de distorsión y posición del centro de distorsión depende de los autores. La función $L(r)$ sólo está definida para valores positivos de r y $L(0) = 1$. Se puede obtener una aproximación a una función de distorsión cualquiera mediante el desarrollo en serie de Taylor $L(r) = 1+\kappa_1 r+ \kappa_2 r^2 + \kappa_3 r^3 + \kappa_4 r^4 + \ldots$. Se considera que los coeficientes de corrección radial $\{\kappa_1, \kappa_2, \kappa_3, \ldots, x_c, y_c\}$ forman parte de los parámetros intrínsecos de la cámara. El valor de los coeficientes κ_i, así como su signo permite corregir los dos tipos de distorsiones geométricas más comunes: de barril y de cojín. A menudo se utiliza el punto principal como centro de la distorsión radial, aunque no tienen por qué coincidir.

Es lícito realizar algunos comentarios sobre el polinomio:

- Término independiente: para $r = 0$ el centro de distorsión no se mueve, por eso el término independiente de $L(r)$ vale 1: $L(0) = 1$.

- Diferenciabilidad en el origen: si el polinomio $L(r)$ incluyese sólo potencias pares (anulamos todos los coeficientes κ_i con i impar), admitiría derivadas continuas de todos los órdenes en el origen, es decir, sería una función $\mathcal{C}^\infty$. Al incluir potencias impares, eso ya no se cumple debido a la no diferenciabilidad de la función raíz cuadrada en el origen. En concreto, una función que incluya κ_1 y κ_2 no nulos es $\mathcal{C}^1$, pero no $\mathcal{C}^2$.

  Parece creíble que el mundo físico varíe de forma suave y que, por tanto, pueda ser descrito mediante funciones de clase $\mathcal{C}^\infty$, lo que nos llevaría a adoptar un modelo de potencias pares en la función de distorsión. Sin embargo, autores como Hartley [1], junto con Devernay y Faugeras adoptan el modelo más general de distorsión radial, el cual incluye también potencias impares, como se comenta en [12].

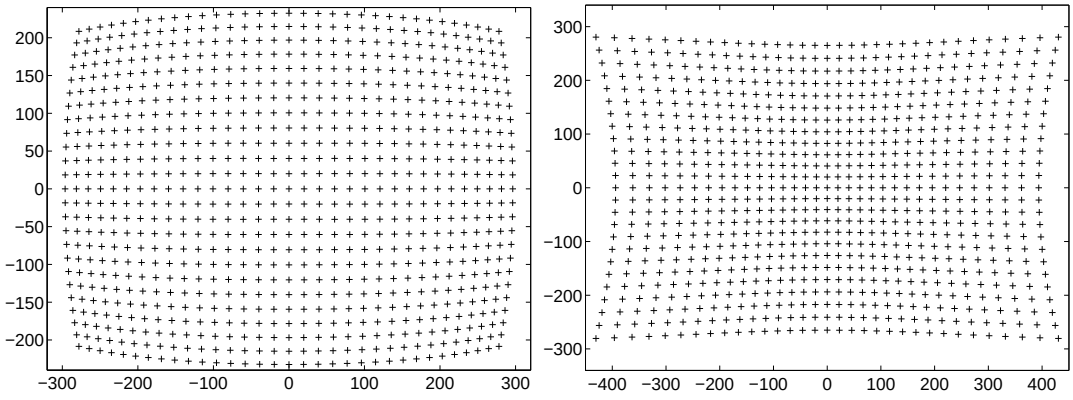


Figura 2.3: Ejemplo de los tipos de distorsiones radiales más frecuentes: distorsión de barril (izquierda) y distorsión de cojín (derecha).

La figura 2.3 ilustra mediante dos simulaciones de MATLAB la distorsión radial: representa la deformación de una malla ortogonal de 640×480 centrada en el origen a través de un modelo de distorsión radial con el centro de distorsión en el origen de coordenadas y un polinomio distorsionador con tres coeficientes: $\kappa_1 = 2 \cdot 10^{-4}$, $\kappa_2 = -1{,}5 \cdot 10^{-6}$ y $\kappa_3 = 4 \cdot 10^{-10}$ para la distorsión de barril y los mismos coeficientes, pero cambiando el signo de κ_2 para la distorsión de cojín.

2.3. Formalización del problema

Con los elementos que ya han sido presentados podemos formalizar el problema más general de la reconstrucción 3D de la forma siguiente. Dadas m imágenes con las proyecciones $\{\mathbf{x}_j^i\}_{i=1,\dots,m;j=1,\dots,n}$ de n puntos (convenientemente identificados uno en cada imagen como correspondientes al mismo punto del espacio), queremos hallar las posiciones de estos puntos en el espacio y los parámetros de las cámaras. Si elegimos arbitrariamente el sistema de referencia euclídeo asociado a una de las imágenes como sistema de referencia, queremos hallar las coordenadas $\{\mathbf{X}_j\}_{j=1,\dots,n}$ de los puntos y las matrices y vectores $\{\mathtt{K}^i, \mathtt{R}^i, \widetilde{\mathbf{C}}^i\}_{i=1,\dots,m}$ tales que

$$\mathbf{x}_j^i \sim \mathtt{K}^i[\mathtt{R}^i| - \mathtt{R}^i\widetilde{\mathbf{C}}^i]\mathbf{X}_j$$

Una limitación esencial es que en la reconstrucción con cámaras no calibradas es imposible recuperar la escala de la escena, puesto que si escalamos las coordenadas de los puntos y los desplazamientos entre cámaras obtenemos las mismas imágenes. La reconstrucción es posible dentro del grupo conforme, es decir, salvo una transformación de semejanza, la cual incluye un escalado, una rotación y una traslación.

Realicemos una cuenta de incógnitas y ecuaciones:

- Incógnitas: Por cada cámara, 5 parámetros intrínsecos y 6 de posición hacen $5m+6(m-1)$, a los que hay que sumar 3 por punto, luego tenemos $5m + 6(m-1) + 3n = 11m + 3n - 6$ incógnitas.
- Ecuaciones: Dos por punto y cámara, es decir, $2mn$.

La presencia de errores en la posición de las proyecciones (ruido en los datos) hace interesante plantear el problema en términos de estimación estadística. Si se supone que el ruido en cada punto proyectado es independiente de los demás y tiene distribución gaussiana de componentes independientes, los parámetros de cámaras y escena más verosímiles serán los que minimicen la suma de los cuadrados de las distancias euclídeas entre los puntos observados $\mathbf{x}_j^i$ y los valores obtenidos a partir de estos parámetros mediante las ecuaciones. Por tanto, un planeamiento natural del problema consiste en la identificarlo con la minimización de la función de coste

$$\sum_{i,j} d^2(\mathbf{x}_j^i, \mathtt{K}^i[\mathtt{R}^i| - \mathtt{R}^i\widetilde{\mathbf{C}}^i]\mathbf{X}_j)$$

donde d representa la distancia euclídea en el plano de la imagen. Esta función de coste se conoce con el nombre de "error de reproyección" y es muy utilizada en todo el proyecto.

Se trata por tanto de un problema de optimización no lineal con un número elevado de variables para el que no es esperable resolver mediante técnicas genéricas de optimización. La estrategia para resolver el problema se basará en dividirlo en subproblemas que se puedan resolver mediante factorización de matrices o problemas de optimización factibles. Posteriormente estos resultados pueden refinarse mediante técnicas de optimización que busquen la minimización de la función de coste anterior o una aproximación a la misma.

2.4. Esquema de procesamiento

El esquema de procesamiento completo desde que se dispone de las imágenes hasta que se visualiza la reconstrucción en tres dimensiones está representado en el diagrama de bloques de la figura 2.4 y consta de los siguientes elementos:

- El primer bloque es el que se encarga de extraer las características de las imágenes y ponerlas en correspondencia. Para una secuencia de vídeo sería el algoritmo de extracción de puntos más el de seguimiento.

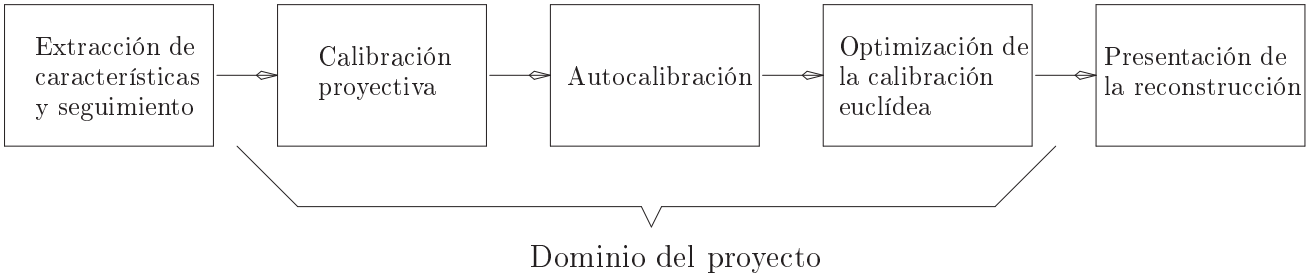


Figura 2.4: Diagrama de bloques del esquema de procesamiento

- Un bloque cuya misión es la calibración proyectiva de las cámaras. Una reconstrucción proyectiva es una reconstrucción de la escena que está relacionada con la reconstrucción real mediante una transformación geométrica del espacio llamada *homografía*.
- El siguiente bloque realiza la autocalibración. Dada una calibración proyectiva de las cámaras, obtiene una homografía para convertir una reconstrucción proyectiva en una reconstrucción métrica (a veces llamada "euclídea").
- Un bloque que optimiza la calibración métrica obtenida, minimizando la función de coste del error de reproyección.
- La presentación de la reconstrucción es el objetivo del último bloque. La información de la calibración euclídea de las cámaras y las posiciones de los puntos 3D se combina con la información de texturas de las imágenes para construir un modelo tridimensional de la superficie del mundo real y visualizarlo mediante algún programa de realidad virtual.

Los bloques pueden estar a su vez divididos en sub-bloques, por ejemplo el bloque de autocalibración puede realizar una calibración estratificada de la escena, en lugar de hacerla en una sola etapa, la ejecuta en varias. Reiteramos, una vez más que los bloques primero y último no forman parte del desarrollo del proyecto.

2.5. Casos particulares

Muchos algoritmos de autocalibración realizan hipótesis sobre los valores de algunos parámetros intrínsecos de las cámaras, en otras ocasiones se imponen restricciones en lugar de conocer esos valores. Algunas de estas suposiciones son:

- Todas las imágenes han sido realizadas con la misma cámara, sin variar ninguno de sus parámetros. Es la hipótesis de parámetros intrínsecos constantes, pero desconocidos. La matriz $\mathtt{K}$ es la misma para todas las cámaras.
- La cámara que adquirió las imágenes posee píxeles rectangulares: $\theta = \pi/2$, (cámara CCD).

$$\mathtt{K} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Los píxeles son cuadrados, es decir, $\theta = \pi/2$ y $\tau = 1 \Leftrightarrow \alpha_u = \alpha_v$. Es equivalente a decir que el ángulo θ y la relación de aspecto son conocidos.

$$\mathtt{K} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_u & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- El punto principal de la cámara es conocido. Es equivalente a decir que el punto principal está en el origen de coordenadas, ya que siempre podemos realizar una traslación de los ejes de tal forma que el punto principal sea el origen del nuevo sistema de coordenadas.

$$\mathtt{K} = \begin{bmatrix} \alpha_u & -\alpha_u \cot\theta & 0 \\ 0 & \alpha_v / \operatorname{sen}\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Algunos algoritmos suponen que se conocen todos los parámetros intrínsecos salvo la distancia focal, es decir, suponen píxeles cuadrados y punto principal en el centro. La matriz de parámetros intrínsecos se simplifica notablemente:

$$\mathtt{K} = \begin{bmatrix} \alpha_u & 0 & 0 \\ 0 & \alpha_u & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Gracias a la experiencia adquirida en pruebas experimentales y el estudio de la literatura sobre autocalibración podemos decir que la hipótesis de píxeles cuadrados es más creíble que la hipótesis de punto principal en el origen. La posición del punto principal puede variar significativamente de una imagen a otra, ambas tomadas con la misma cámara; sin embargo la relación de aspecto y el *skew* son parámetros fijos debidos a la construcción del dispositivo CCD de las cámaras digitales. Éste no cambia de una imagen a otra.

Capítulo 3

Geometría Proyectiva

3.1. Introducción

La geometría euclídea describe localmente nuestro mundo tridimensional. Todos estamos familiarizados con esta geometría ya que desde pequeños se nos enseña en el colegio. En la geometría euclídea los lados de los objetos tienen longitudes, las líneas que se intersecan determinan ángulos entre ellas, y se dice que dos líneas son paralelas si pertenecen al mismo plano y nunca se cortan. Además, estas propiedades no cambian cuando aplicamos las transformaciones euclídeas (traslación y rotación). Sin embargo, no es el único tipo de geometría. Esto queda de manifiesto cuando consideramos la formación de imágenes en una cámara, la geometría euclídea es insuficiente: no se preservan las longitudes ni los ángulos, y las líneas paralelas se pueden cortar. Nota: utilizamos el término "transformación" para referirnos a las biyecciones.

	Euclídea	Semejanza	Afín	Proyectiva
Transformaciones				
rotación	X	X	X	X
traslación	X	X	X	X
escalado uniforme		X	X	X
escalado no uniforme			X	X
deformación/inclinación			X	X
homografías				X
Invariantes				
longitud	X			
ángulo	X	X		
proporciones entre segmentos	X	X		
razón simple	X	X	X	
paralelismo	X	X	X	
incidencia	X	X	X	X
razón doble	X	X	X	X

Cuadro 3.1: Las cuatro geometrías diferentes, las transformaciones permitidas en cada una y las medidas que permanecen invariantes bajo esas transformaciones

La geometría euclídea es en realidad un subconjunto de lo que se conoce como *geometría proyectiva*. De hecho, hay dos geometrías entre ambas: la geometría del grupo conforme formado por las *semejanzas* y la geometría *afín*. La relación entre las diferentes geometrías la podemos consultar en la tabla 3.1, que se irá comprendiendo a medida que se avance en la lectura de este capítulo.

La geometría proyectiva modela bien el proceso de adquisición de imágenes en una cámara porque permite una clase más amplia de transformaciones que sólo traslaciones y rotaciones. También permite tratar más adecuadamente la proyección cónica. Las ventajas de las transformaciones proyectivas son: preservan el tipo (esto es, los puntos siguen siendo puntos y las rectas se transforman en rectas), la incidencia (esto es, si un punto pertenece a una recta) y una medida conocida como la *razón doble*. La desventaja es que se preservan menos medidas: ni longitudes, ni ángulos, ni paralelismo.

La geometría proyectiva existe para cualquier número de dimensiones, al igual que la geometría euclídea. Por ejemplo la recta proyectiva, que denotamos por $\mathbb{P}^1$, es análoga al mundo euclídeo unidimensional; el plano proyectivo, $\mathbb{P}^2$, se corresponde con el plano euclídeo; y el espacio proyectivo, $\mathbb{P}^3$, está vinculado con el espacio euclídeo tridimensional. El proceso de formación de la imagen es una proyección desde $\mathbb{P}^3$ hasta $\mathbb{P}^2$. Como es más fácil comprender los conceptos principales en espacios de pequeña dimensión, haremos un gran esfuerzo en estudiar $\mathbb{P}^2$, el plano proyectivo. Este espacio es especialmente útil para entender el plano de la imagen y tiene conceptos análogos en $\mathbb{P}^3$.

Si se desea profundizar más en la geometría proyectiva se recomiendan las referencias [13] y [14].

3.2. El Plano Proyectivo

3.2.1. Cuatro modelos

Hay cuatro formas de pensar en el plano proyectivo [34]. La más importante de ellas para la aplicación a la visión artificial es la de coordenadas homogéneas. Empezando por el modelo de coordenadas homogéneas y prosiguiendo con cada uno de los otros tres modelos trataremos de obtener una intuición de la naturaleza del plano proyectivo, cuya definición concisa proviene del cuarto modelo.

Coordenadas homogéneas

Supongamos que tenemos un punto $(x, y)^\top$ en el plano euclídeo. Como ya hemos visto, para representar este mismo punto en el plano proyectivo, simplemente añadimos una tercera coordenada de valor 1 al final: $(x, y, 1)^\top$. Esta operación recibe el nombre de homogeneización de las coordenadas. En general, un punto en un espacio euclídeo n-dimensional se representa por un punto en un espacio proyectivo de dimensión $(n + 1)$. La igualdad es sustituida por la proporcionalidad: el punto $(x, y, 1)^\top$ es el mismo que el punto $(\alpha x, \alpha y, \alpha)^\top$ para cualquier λ no nulo.

$$(X, Y, W)^\top = (\lambda X, \lambda Y, \lambda W)^\top \qquad \forall \lambda \neq 0$$

El punto $(0, 0, 0)$ no está permitido. Las coordenadas se llaman homogéneas porque el factor de proporcionalidad no es importante.

Para representar una recta en el plano proyectivo partimos de la fórmula euclídea de la recta

$$ax + by + c = 0$$

y ya que la ecuación no se ve afectada por el factor de proporcionalidad llegamos a lo siguiente:

$$aX + bY + cW = 0 \tag{3.1}$$
$$\mathbf{u}^\top \mathbf{p} = \mathbf{p}^\top \mathbf{u} = 0 \tag{3.2}$$

donde $\mathbf{u} = [a, b, c]^\top$ es la recta y $\mathbf{p} = [X, Y, W]^\top$ es el punto sobre la recta. Vemos que los puntos y las rectas tienen la misma representación en el plano proyectivo. Los parámetros de una recta se pueden interpretar fácilmente: $-a/b$ es la pendiente, $-c/a$ es el punto de corte con el eje x, y $-c/b$ es el punto de corte con el eje y.

Para transformar un punto en el plano proyectivo de vuelta a coordenadas euclídeas, simplemente hay que dividir por la tercera coordenada: $(x, y)^\top = (X/W, Y/W)^\top$. Esta operación recibe el nombre de deshomogeneización de las coordenadas. Inmediatamente vemos que el plano proyectivo contiene más puntos que el plano euclídeo: los puntos cuya tercera coordenada es cero. Son los llamados puntos del infinito y representan las direcciones de todas las rectas del plano euclídeo. Hay un punto del infinito distinto asociado con cada dirección en el plano; por ejemplo, los puntos $(1, 0, 0)^\top$ y $(0, 1, 0)^\top$ están asociados con las direcciones horizontal y vertical, respectivamente. Los puntos del infinito son considerados como cualquier otro punto en $\mathbb{P}^2$ y no se les aplica ningún trato especial. Todos los puntos del infinito están sobre una recta, llamada la recta del infinito, que una vez más, es tratada de la misma forma que cualquier otra recta. La recta del infinito canónica en un sistema de referencia euclídeo tiene la representación $(0, 0, 1)^\top$.

Supongamos que queremos encontrar la intersección de dos rectas. Por álgebra elemental, las rectas $\mathbf{u}_1 = (a_1, b_1, c_1)^\top$ y $\mathbf{u}_2 = (a_2, b_2, c_2)^\top$ se cortan en el punto $\mathbf{p} = (b_1c_2 - b_2c_1, a_2c_1 - a_1c_2, a_1b_2 - a_2b_1)^\top$. Esta fórmula es fácilmente recordable como el producto vectorial: $\mathbf{p} = \mathbf{u}_1 \times \mathbf{u}_2$. Si las dos rectas son paralelas, $-a_1/b_1 = -a_2/b_2$, el punto de corte es $(b_1c_2 - b_2c_1, a_2c_1 - a_1c_2, 0)$, que es el punto del infinito asociado con la dirección cuya pendiente es $-a_1/b_1$. De forma similar, dados dos puntos $\mathbf{p}_1$ y $\mathbf{p}_2$, la ecuación de la recta que pasa por ambos viene dada por $\mathbf{u} = \mathbf{p}_1 \times \mathbf{p}_2$.

Ahora supongamos que queremos determinar si tres puntos $\mathbf{p}_1$, $\mathbf{p}_2$ y $\mathbf{p}_3$ están alineados (sobre la misma recta). La recta que une los dos primeros puntos es $\mathbf{p}_1 \times \mathbf{p}_2$. El tercer punto está sobre la recta si $\mathbf{p}_3^\top(\mathbf{p}_1 \times \mathbf{p}_2) = 0$, es decir, si el determinante de la matriz de 3×3 que contiene las coordenadas de los puntos es cero:

$$\det[\mathbf{p}_1\ \mathbf{p}_2\ \mathbf{p}_3] = 0.$$

De forma similar, tres líneas $\mathbf{u}_1$, $\mathbf{u}_2$ y $\mathbf{u}_3$ se cortan en al mismo punto (son concurrentes) si se cumple la siguiente ecuación:

$$\det[\mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3] = 0.$$

En la tabla 3.2 resumimos los conceptos de coordenadas homogéneas.

punto	$\mathbf{p} = (X, Y, W)^\top$
incidencia	$\mathbf{p}^\top\mathbf{u} = 0$
alineación	$\det[\mathbf{p}_1\ \mathbf{p}_2\ \mathbf{p}_3] = 0$
recta que une dos puntos	$\mathbf{u} = \mathbf{p}_1 \times \mathbf{p}_2$
punto del infinito	$(X, Y, 0)^\top$

recta	$\mathbf{u} = (a, b, c)^\top$
incidencia	$\mathbf{p}^\top\mathbf{u} = 0$
concurrencia	$\det[\mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3] = 0$
punto intersección de dos rectas	$\mathbf{p} = \mathbf{u}_1 \times \mathbf{u}_2$
recta del infinito	$(0, 0, c)^\top$

Cuadro 3.2: Resumen de las coordenadas homogéneas en $\mathbb{P}^2$: puntos y rectas

Ejemplo 1: Dadas dos rectas $\mathbf{u}_1 = (4, 2, 2)$ y $\mathbf{u}_2 = (6, 5, 1)$, el punto de corte viene dado por:

$$\begin{vmatrix} i & j & k \\ 4 & 2 & 2 \\ 6 & 5 & 1 \end{vmatrix} = (2 - 10)i + (12 - 4)j + (20 - 12)k = (-8, 8, 8) = (-1, 1, 1)$$

Ejemplo 2: Consideremos la intersección de la hipérbola $xy = 1$ con la recta horizontal $y = 1$. Para convertir estas ecuaciones a coordenadas homogéneas recordemos que $X = Wx$ y $Y = Wy$, lo que da $XY = W^2$ para la hipérbola y $Y = W$ para la recta. La solución a estas dos ecuaciones es el punto (W, W, W), que es el mismo que el punto $(1, 1)$ en el plano euclídeo, el resultado deseado. Ahora consideremos la intersección de la misma parábola con la recta horizontal $y = 0$, una intersección que no existe en el plano euclídeo. En coordenadas homogéneas la recta es $Y = 0$ lo que produce la solución $(X, 0, 0)$, el punto del infinito asociado con la dirección horizontal.

Espacio de rayos

Acabamos de ver que, al pasar de lo euclídeo a lo proyectivo, un punto en $\mathbb{R}^2$ se convierte en un conjunto de puntos en $\mathbb{R}^3$, que están relacionados por un factor de escala no nulo. Así pues, un punto $\mathbf{p} = (X, Y, W)$ en $\mathbb{P}^2$ se puede ver como una "recta" en el espacio tridimensional que pasa por el origen y el punto $\mathbf{p}$ (técnicamente hablando, la recta no incluye el origen. Este espacio tridimensional es conocido como el espacio de rayos (entre otros nombres) y una representación es la figura 3.1. Del mismo modo, una recta $\mathbf{u} = (a, b, c)$ en $\mathbb{P}^2$ se puede ver como un "plano" que pasa por el origen y es perpendicular a $\mathbf{u}$. La recta del infinito es el "plano" horizontal $W = 0$, y los puntos del infinito son "rectas" en este "plano".

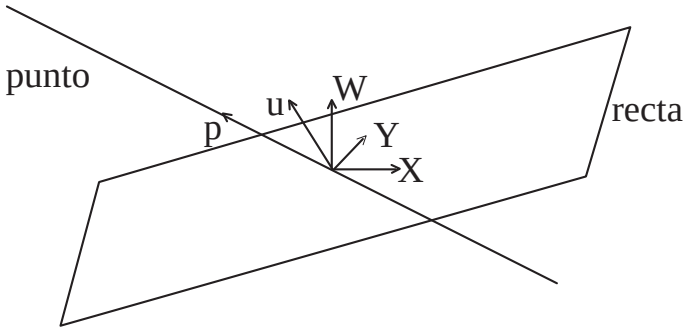


Figura 3.1: Espacio de rayos

La esfera unidad

Debido a que las coordenadas no se ven afectadas por un factor de proporcionalidad, $\mathbb{P}^2$ es bidimensional, aunque sus puntos tengan tres coordenadas. Cada punto $\mathbf{p} = (X, Y, W)$, representado por una "recta" en el espacio de rayos, puede ser proyectado sobre la esfera unidad para obtener el punto $\frac{1}{\sqrt{X^2+Y^2+Z^2}}(X, Y, W)$. El denominador nunca es cero porque el punto $(0, 0, 0)$ no está permitido.

Los puntos en el plano proyectivo pueden verse como puntos en la esfera unidad, como muestra la figura 3.2. Como cada "recta" en el espacio de rayos corta a la esfera en dos puntos, ambas intersecciones representan el mismo punto: los puntos antipodales son idénticos. Así mismo, los "planos" que representan rectas en el espacio de rayos cortan a la esfera unidad en grandes circunferencias (meridianos), así las rectas se pueden ver como grandes circunferencias perpendiculares a $\mathbf{u}$. La recta del infinito es el ecuador de la esfera, y los puntos del infinito están sobre esta circunferencia.

A veces se utiliza una jerga matemática que va un poco más allá y se dicen frases como: "$\mathbb{S}^2$ es una doble cubierta de $\mathbb{P}^2$". De forma simplificada esto significa que *localmente* existe una biyección entre la esfera y el plano proyectivo, si nos olvidamos de los puntos antipodales.

El plano afín aumentado

Ahora, proyectemos la esfera unidad sobre el plano $W = 1$. Cada punto (X, Y, W) de la esfera es transformado en el punto $(\frac{X}{W}, \frac{Y}{W}, 1)$ que está en la intersección del plano $W = 1$ con la "recta" que representa el punto. Del mismo modo, las rectas son convertidas en la intersección del plano $W = 1$ con el "plano" que representa a la recta. Los puntos del infinito y la recta del infinito se proyectan en puntos del infinito y la recta del infinito, respectivamente, como muestra la figura 3.3. Volvemos a una representación en la que los puntos son puntos y las rectas son rectas.
El plano proyectivo, $\mathbb{P}^2$, es el plano afín aumentado por una sola recta del infinito y un conjunto de puntos del infinito, uno para cada dirección. La recta del infinito y los puntos del infinito no se tratan de forma distinta a la recta y puntos finitos. El plano afín contiene los mismos puntos que el plano

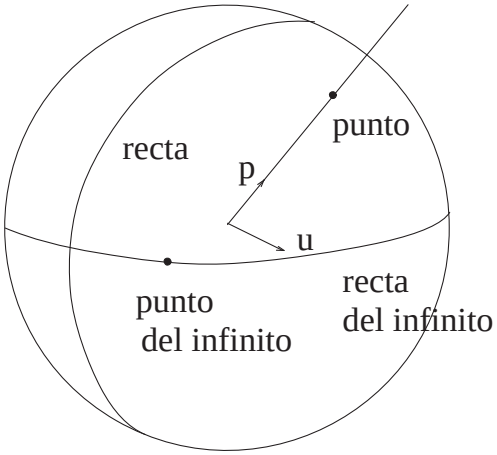


Figura 3.2: La esfera unidad

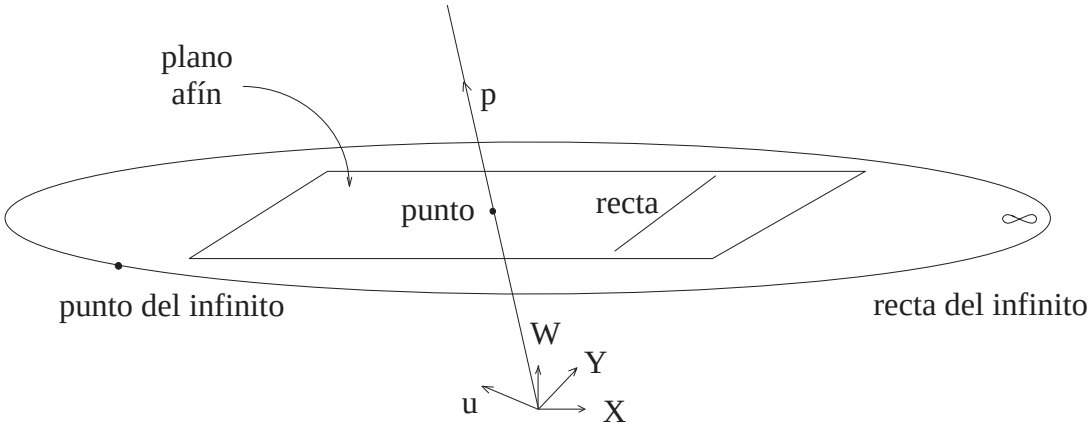


Figura 3.3: El plano afín más la recta del infinito y los puntos del infinito

euclídeo. La única diferencia es que el primero también permite escalado no uniforme y deformación (inclinación).

3.2.2. Dualidad

Observando una vez más la tabla 3.2, las semejanzas entre puntos y rectas son sorprendentes. Por ejemplo, sus representaciones son idénticas, y la fórmula para la intersección de dos rectas es la misma que la fórmula para la recta que pasa por dos puntos. Estas observaciones no son un resultado de la coincidencia, sino un resultado de la dualidad que existe entre puntos y rectas en el plano proyectivo. En otras palabras, cualquier teorema o sentencia que es verdadero para el plano proyectivo puede ser reescrito intercambiando puntos por rectas y alineación por incidencia; el teorema resultante también es cierto.

Haz de rectas

Un conjunto de rectas concurrentes (rectas que pasan por un mismo punto) en $\mathbb{P}^2$ es un espacio proyectivo unidimensional llamado haz de rectas. Éste es un resultado obvio de aplicar el principio de dualidad: un conjunto de rectas concurrentes es lo mismo que un conjunto de puntos alineados.

3.2.3. La razón doble

El invariante de la geometría proyectiva es la razón doble, que es una razón de proporciones asociada a cuaternas de puntos alineados de un espacio proyectivo. Dados cuatro puntos alineados $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$ y $\mathbf{p}_4$ de $\mathbb{P}^2$, no necesitamos las tres coordenadas para expresar su localización ya que una de las tres coordenadas depende de las otras dos según la ecuación de la recta sobre la que están. Así que podemos eliminar una de sus coordenadas, ya que la sabremos recuperar a través de la dependencia que establece la ecuación de la recta. Supongamos que dichas coordenadas son $(p_{i1}, p_{i2}), i = 1, \ldots, 4$, la razón doble se calcula de la siguiente forma:

$$Cr(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) = \frac{\Delta_{31}\Delta_{42}}{\Delta_{32}\Delta_{41}}, \quad \text{donde} \quad \Delta_{ij} = \begin{vmatrix} p_{i1} & p_{i2} \\ p_{j1} & p_{j2} \end{vmatrix}$$

La razón doble es independiente de la elección de coordenadas que se realice sobre la recta en que están los puntos. Aunque la razón doble es un invariante una vez que elegimos el orden de los puntos, su valor depende de ese orden. Podemos ordenar cuatro puntos de $4! = 24$ formas distintas, pero en realidad sólo se obtienen como máximo seis valores distintos de la razón doble, que están descritos por el conjunto

$$\{\rho, \frac{1}{\rho}, 1-\rho, \frac{1}{1-\rho}, \frac{\rho-1}{\rho}, \frac{\rho}{\rho-1}\}.$$

Hay otras medidas de la razón doble y todas ellas son invariantes bajo transformaciones proyectivas. La dualidad conduce a una razón doble para cuatro rectas concurrentes al sustituir la distancia euclídea entre dos puntos por el seno del ángulo entre dos rectas. Otra forma menos obvia de medir la razón doble entre cuatro rectas concurrentes es utilizar otra recta, arbitraria y que corte a las cuatro anteriores; la razón doble de las rectas la definimos como la razón doble de los cuatro puntos de intersección (la razón doble es la misma sin importar qué recta se escoja).

3.2.4. Referencia proyectiva

Una referencia proyectiva de un espacio $\mathbb{P}^n$ es una colección ordenada de $n+2$ puntos $x_0, \ldots, x_n, x_{n+1}$ de $\mathbb{P}^n$ tal que cualesquiera $n+1$ entre ellos son independientes. El último de los puntos se denomina punto unidad. Se denota:

$$\mathcal{R} = \{x_0, \ldots, x_n; x_{n+1}\}.$$

Así como una referencia euclídea del plano está formada por un punto origen y dos vectores (3 puntos), una referencia del plano proyectivo está formada por 4 puntos. Las transformaciones propias de cada espacio se definen de acuerdo a a cómo transformen sus referencias de origen y destino.

3.2.5. Cónicas y cónicas duales

En geometría euclídea, las cónicas de segundo orden (elipses, parábolas, hipérbolas, etc.) son objetos geométricos importantes. Colectivamente, nos referimos a estas curvas como cónicas, sin distinguir entre las distintas formas euclídeas.

Así como una circunferencia en geometría euclídea se define como el lugar de los puntos que equidistan del centro, una cónica en geometría proyectiva se define como el lugar de los puntos con razón doble constante a cuatro puntos fijos, de los cuales no hay tres alineados. En ambos casos la forma de la curva se define respecto a un invariante de la geometría particular: distancia en el caso de geometría euclídea, y razón doble en el caso proyectivo.

La ecuación de la cónica viene dada por:

$$\mathbf{p}^\top C \mathbf{p} = 0,$$

o

$$c_{11}X^2 + c_{22}Y^2 + c_{33}W^2 + 2c_{12}XY + 2c_{13}XW + 2c_{23}YW = 0$$

donde $\mathbf{p}$ es un vector columna de tres componentes (3×1) y C es una matriz simétrica de 3×3.

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & c_{23} \\ c_{13} & c_{23} & c_{33} \end{bmatrix}$$

La matriz C es una representación homogénea de una cónica. Por lo tanto, de los seis grados de libertad iniciales de una matriz simétrica de 3×3 hay que quitar uno, debido a que el factor de proporcionalidad no es importante: multiplicar C por una constante no altera la ecuación de la cónica $\mathbf{p}^\top C\mathbf{p} = 0$. Por lo tanto, tiene cinco grados de libertad.

Cinco puntos definen una cónica

Supongamos que queremos calcular la cónica que pasa por un conjunto de puntos $\mathbf{p}_i$. ¿Cuántos puntos determinan la cónica unívocamente? Podemos responder a la pregunta dando un algoritmo que calcule la cónica [1]. Cada punto de la cónica verifica $\mathbf{p}_i^\top C\mathbf{p}_i = 0$. Esto se puede poner como un producto matricial, si llamamos

$$\mathbf{q}_i = \begin{bmatrix} X_i^2 & 2X_iY_i & 2X_iW_i & Y_i^2 & 2Y_iW_i & W_i^2 \end{bmatrix}^\top$$

entonces cada punto genera la condición

$$\mathbf{q}_i^\top \mathbf{c} = 0$$

donde $\mathbf{c} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{22} & c_{23} & c_{33} \end{bmatrix}^\top$ es la cónica representada por un vector de 6×1. El resultado de poner cinco condiciones de puntos es un sistema de ecuaciones lineales en los elementos de C.

$$\begin{bmatrix} \mathbf{q}_1^\top \\ \mathbf{q}_2^\top \\ \mathbf{q}_3^\top \\ \mathbf{q}_4^\top \\ \mathbf{q}_5^\top \end{bmatrix} \mathbf{c} = 0$$

y la cónica es el vector solución del sistema de 5 ecuaciones y 6 incógnitas: el vector que expande el núcleo (de dimensión uno) de la matriz del sistema.

Rectas tangentes a una cónica.La recta $\mathbf{l}$ tangente a la cónica C en el punto $\mathbf{p}$ es, en coordenadas homogéneas: $\mathbf{l} = C\mathbf{p}$.

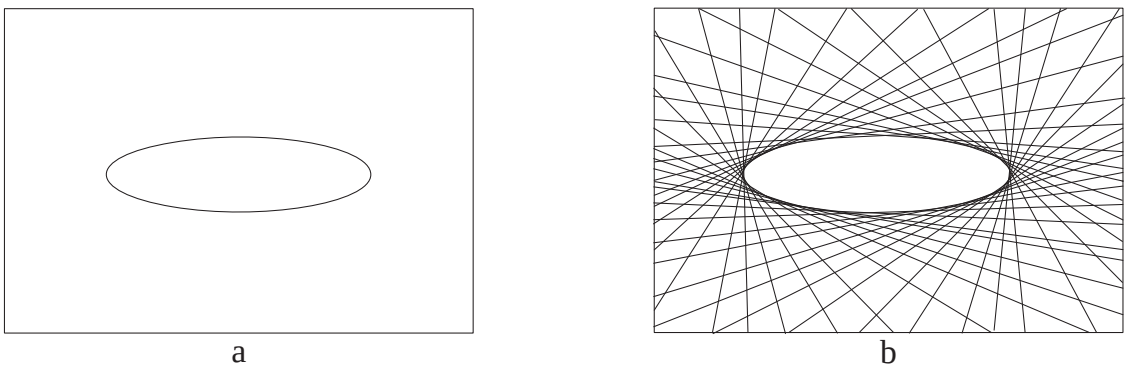


Figura 3.4: (a) Los puntos $\mathbf{p}$ que verifican $\mathbf{p}^\top C\mathbf{p} = 0$ están sobre una cónica de puntos o cónica puntual. (b) Las rectas $\mathbf{l}$ que cumplen $\mathbf{l}^\top C^*\mathbf{l} = 0$ son tangentes a la cónica de puntos C. La cónica C es la envolvente de las rectas $\mathbf{l}$.

Cónicas duales

La cónica C definida anteriormente es llamada con mayor propiedad una cónica *de puntos*, porque define una ecuación para puntos. Dado el principio de dualidad en $\mathbb{P}^2$ entre puntos y rectas no es sorprendente que también exista una cónica que define una ecuación para rectas. Esta cónica *dual* (o cónica de rectas) también se representa por una matriz de 3×3, que denotamos como C^*. Una recta $\mathbf{l}$ tangente a la cónica C satisface $\mathbf{l}^\top C^* \mathbf{l} = 0$. La notación C^* indica que C^* es la matriz adjunta de C (definida a través de la inversa: $C^* = \det(C)C^{-\top}$). Para matrices regulares y simétricas, $C^* = C^{-1}$ (salvo proporcionalidad).

La ecuación de una cónica dual es fácil de obtener en el caso de que C tenga rango completo: en un punto $\mathbf{p}$ sobre C, la tangente es $\mathbf{l} = C\mathbf{p}$. Dándole la vuelta, vemos que el punto $\mathbf{p}$ en el que la recta $\mathbf{l}$ es tangente a C es $\mathbf{p} = C^{-1}\mathbf{l}$. Como $\mathbf{p}$ satisface $\mathbf{p}^\top C\mathbf{p} = 0$, obtenemos que $(C^{-1}\mathbf{l})^\top C(C^{-1}\mathbf{l}) = \mathbf{l}^\top C^* \mathbf{l} = 0$, sustituyendo en el último paso $C^{-\top} = C^{-1}$ porque C es simétrica.

Una cónica dual tiene cinco grados de libertad. Del mismo modo que cinco puntos cualesquiera definen una cónica de puntos, cinco rectas cualesquiera determinan una cónica dual. En la figura 3.4 vemos una representación de una cónica de puntos y su dual.

3.2.6. Puntos circulares

Una sorprendente propiedad de las cónicas es que cualquier circunferencia corta a la recta del infinito, $W = 0$, en dos puntos fijos. Para ver esto, notemos que una circunferencia es una cónica con matriz diagonal y cuyos elementos de la diagonal principal son iguales $(1, 1, -1)$:

$$X^2 + Y^2 - W^2 = 0,$$

que corta a la recta del infinito $W = 0$ en

$$X^2 + Y^2 = 0$$

Esta ecuación tiene dos raíces complejas, conocidas como los puntos circulares: $\mathbf{I} = (1, i, 0)$ y $\mathbf{J} = (1, -i, 0)$. (Aunque, por simplicidad hemos asumido coordenadas homogéneas reales, estas pueden ser, en general, elementos de cualquier cuerpo conmutativo en el que $1 + 1 \neq 0$ [32]). Los puntos circulares permanecen invariantes bajo transformaciones de semejanza, lo que los hace útiles para determinar los ángulos entre dos rectas, mediante la fórmula de Laguerre.

3.2.7. Homografías

Las homografías son las biyecciones propias de los espacios proyectivos. Una homografía de $\mathbb{P}^2$ se define como una aplicación del plano en sí mismo, tal que la alineación de cualquier conjunto de puntos se mantiene. Tal aplicación se consigue con la multiplicación por una matriz $\mathtt{T}$ de 3×3. Cada punto $\mathbf{p}$ se transforma en otro punto $\mathbf{p}'$:

$$\mathbf{p}' = \mathtt{T}\mathbf{p}$$

Utilizaremos los términos transformación y homografía casi indistintamente. Salvo proporcionalidad, sólo ocho términos de la matriz $\mathtt{T}$ son independientes (hay ocho grados de libertad). Como cada punto contiene dos valores independientes, para determinar $\mathtt{T}$ basta con tener cuatro parejas de puntos correspondientes.

Para convertir una recta $\mathbf{u}$ en otra recta $\mathbf{u}'$ necesitamos que se preserve la incidencia, esto es, si un punto $\mathbf{p}$ está sobre la recta $\mathbf{u}$, entonces el punto correspondiente $\mathbf{p}'$ debe estar sobre la correspondiente recta $\mathbf{u}'$. Por lo tanto,

$$\mathbf{p}^\top \mathbf{u} = 0 = (\mathtt{T}^{-1}\mathbf{p}')^\top \mathbf{u} = \mathbf{p}'^\top (\mathtt{T}^{-\top}\mathbf{u}),$$

lo que indica que si los puntos se transforman según $\mathtt{T}$, los elementos duales (las rectas en $\mathbb{P}^2$) se transforman según $\mathtt{T}^{-\top}$.

$$\mathbf{u}' = \mathtt{T}^{-\top}\mathbf{u}$$

A partir de estos resultados no es difícil de demostrar que una cónica puntual C se transforma en $\mathtt{T}^{-\top}C\mathtt{T}^{-1}$, y una cónica de rectas $C^* = |C|C^{-1}$ se transforma en $\mathtt{T}C^*\mathtt{T}^\top$.

DEMO: Los puntos $\mathbf{p}$ se transforman de acuerdo a $\mathbf{p}' = \mathtt{T}\mathbf{p} \Leftrightarrow \mathbf{p} = \mathtt{T}^{-1}\mathbf{p}'$. Sustituyendo en la ecuación de la cónica, $0 = \mathbf{p}^\top C\mathbf{p} = (\mathtt{T}^{-1}\mathbf{p}')^\top C\mathtt{T}^{-1}\mathbf{p}' = \mathbf{p}'^\top(\mathtt{T}^{-\top}C\mathtt{T}^{-1})\mathbf{p}' = \mathbf{p}'^\top C'\mathbf{p}'$, es decir, $C \mapsto C' = \mathtt{T}^{-\top}C\mathtt{T}^{-1}$.

Las rectas se transforman según $\mathbf{u}' = \mathtt{T}^{-\top}\mathbf{u} \Leftrightarrow \mathbf{u} = \mathtt{T}^\top\mathbf{u}'$ y verifican la ecuación de la cónica $0 = \mathbf{u}^\top C^*\mathbf{u} = (\mathtt{T}^\top\mathbf{u}')^\top C^*\mathtt{T}^\top\mathbf{u}' = \mathbf{u}'^\top(\mathtt{T}C^*\mathtt{T}^\top)\mathbf{u}' = \mathbf{u}'^\top C'^*\mathbf{u}'$, o sea, $C^* \mapsto C'^* = \mathtt{T}C^*\mathtt{T}^\top$.

Jerarquía de transformaciones

En cuanto a transformaciones, recordemos que *proyectivo* $\supset$ *afín* $\supset$ *semejanza* $\supset$ *euclídeo*. Estudiemos la matriz T para descubrir las relaciones entre estas geometrías. Primero escribamos los elementos de la matriz proyectiva, por referencia:

$$\mathtt{T}_\mathrm{P} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$$

El plano afín es el plano proyectivo sin la recta del infinito. Por lo tanto, las transformaciones afines deben preservar la recta del infinito y los puntos del infinito, es decir, cualquier punto $[X, Y, 0]^\top$ debe ser transformado en el punto $[\alpha X, \alpha Y, 0]^\top$, para algún escalar arbitrario α:

$$\alpha[X, Y, 0]^\top = \mathtt{T}[X, Y, 0]^\top$$

Lo que implica que $t_{31} = t_{32} = 0$. Por lo que la matriz para transformaciones afines es:

$$\mathtt{T}_\mathrm{A} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ 0 & 0 & t_{33} \end{bmatrix} \sim \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

donde una vez más sólo seis de estos parámetros son independientes, al descontar la proporcionalidad. Estos seis grados de libertad de una transformación afín se reparten en: dos para la traslación $\mathbf{t}$, otros dos para el escalado, uno para un ángulo de rotación θ y otro para un ángulo de inclinación o deformación.

Al contrario que las transformaciones afines, las transformaciones de semejanza preservan ángulos y las proporciones (razones entre longitudes). El resultado es que la matriz de una transformación de semejanza es:

$$\mathtt{T}_\mathrm{S} = \begin{bmatrix} \cos\theta & \sin\theta & t_{13} \\ -\sin\theta & \cos\theta & t_{23} \\ 0 & 0 & t_{33} \end{bmatrix} \sim \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

donde θ es un ángulo arbitrario que indica una rotación, de matriz:

$$\mathtt{R} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Una matriz de transformación de semejanza tiene 4 grados de libertad: uno, θ, para la rotación, uno para el escalado s y dos para la traslación $\mathbf{t}$.

En las transformaciones euclídeas, el factor proporcionalidad sí es importante, $s = 1$: el punto $\mathbf{p}$ se debe convertir a coordenadas euclídeas dividiendo por la tercera coordenada antes de aplicar la transformación:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \Rightarrow \mathtt{T}_{\mathrm{E}} \sim \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$$

3.3. El Espacio Proyectivo

Todos los conceptos discutidos para el plano proyectivo, $\mathbb{P}^2$, tienen analogías en el espacio proyectivo, $\mathbb{P}^3$. Por ejemplo, hay una dualidad entre puntos y planos, las rectas son auto-duales, un haz de planos es un espacio proyectivo bidimensional, la razón doble entre dos planos es invariante, las cuádricas juegan el mismo papel que las cónicas, la cónica absoluta permanece invariante bajo transformaciones de semejanza, y la fórmula de Laguerre puede ser utilizada para encontrar el ángulo entre dos rayos de proyección.

Un punto en $\mathbb{P}^3$ se representa por una 4-tupla $\mathbf{X} = (X, Y, Z, W)^\top$, y lo mismo para un plano $\boldsymbol{\pi}$. Un punto pertenece a un plano si y sólo si $\boldsymbol{\pi}^\top \mathbf{X} = 0$.

Plano que pasa por tres puntos. Tres puntos $\mathbf{X}_i, i = 1, \ldots, 3$, en posición general definen un único plano $\boldsymbol{\pi}$. Cada punto debe verificar $\boldsymbol{\pi}^\top \mathbf{X}_i = 0$, en forma de matriz,

$$\begin{bmatrix} \mathbf{X}_1^\top \\ \mathbf{X}_2^\top \\ \mathbf{X}_3^\top \end{bmatrix} \boldsymbol{\pi} = \mathbf{0}.$$

La matriz de 3×4 cuyas filas son los puntos $\mathbf{X}_i$ tiene rango 3, así que su núcleo, de dimensión 1, determina el plano $\boldsymbol{\pi}$.

Punto intersección de tres planos. Del mismo modo se determina el único punto de intersección $\mathbf{X}$ de tres planos $\boldsymbol{\pi}_i, i = 1, \ldots, 3$ en posición general, ya que son problemas duales. El punto $\mathbf{X}$ es el vector del núcleo de la matriz de 3×4 cuyas filas son las coordenadas homogéneas de los planos $\boldsymbol{\pi}_i$.

$$\begin{bmatrix} \boldsymbol{\pi}_1^\top \\ \boldsymbol{\pi}_2^\top \\ \boldsymbol{\pi}_3^\top \end{bmatrix} \mathbf{X} = \mathbf{0}.$$

3.3.1. Cuádricas

Una cuádrica es una superficie de $\mathbb{P}^3$ que representa una ecuación de segundo grado en cuatro variables homogéneas. No se profundizará sobre la clasificación de las cuádricas, sólo diremos que existen cuádricas de puntos y sus duales: las cuádricas de planos.

Los puntos que pertenecen a la cuádrica verifican la ecuación de segundo grado $\mathbf{X}^\top \mathtt{Q} \mathbf{X} = 0$, siendo $\mathtt{Q}$ la matriz simétrica 4×4 que representa la cuádrica. Las cuádricas duales tienen una expresión análoga, sustituyendo puntos por planos: $\boldsymbol{\pi}^\top \mathtt{Q}^* \boldsymbol{\pi} = 0$, donde $\mathtt{Q}^* \sim \mathtt{Q}^{-1}$, siguiendo el mismo desarrollo que se hizo con las cónicas duales.

Una cuádrica tiene 9 grados de libertad: los mismos que elementos de una matriz simétrica de 4×4 definida salvo proporcionalidad. Así que nueve puntos en posición general determinan una cuádrica de $\mathbb{P}^3$. Si la matriz que define la cuádrica es degenerada, entonces basta con menos puntos para determinarla.

3.3.2. Representación de rectas: las coordenadas de Plücker

Las rectas de $\mathbb{P}^3$ tienen 4 grados de libertad y admiten varias representaciones algebraicas; las más habituales son una matriz antisimétrica de 4×4 o un vector de seis componentes. Ambas están estrechamente relacionadas. Una recta se puede definir de varias maneras, por ejemplo: la recta que pasa por dos puntos dados o la recta intersección de dos planos cualesquiera.

Recta que pasa por dos puntos. La recta que pasa por dos puntos $\mathbf{A}$ y $\mathbf{B}$ se representa por la matriz $\mathtt{L} = \mathbf{A}\mathbf{B}^\top - \mathbf{B}\mathbf{A}^\top$, cuyos elementos son $l_{ij} = A_i B_j - B_i A_j$.

$$\mathtt{L} = \begin{bmatrix} 0 & l_{12} & l_{13} & l_{14} \\ -l_{12} & 0 & l_{23} & l_{24} \\ -l_{13} & -l_{23} & 0 & l_{34} \\ -l_{14} & -l_{24} & -l_{34} & 0 \end{bmatrix}.$$

- Esta matriz $\mathtt{L}$ es antisimétrica y de rango 2. Su núcleo, de dimensión 2, está formado por dos planos del haz de planos cuya base es la recta.
- La matriz $\mathtt{L}$ tiene 6 elementos distintos, pero está definida salvo proporcionalidad y verifica la restricción $\det \mathtt{L} = 0$, que es lo que se conoce como pertenencia de las rectas de $\mathbb{P}^3$ a la cuádrica de Klein de $\mathbb{P}^5$.

$$\det \mathtt{L} = (l_{12}l_{34} - l_{13}l_{24} + l_{14}l_{23})^2 = 0 \Leftrightarrow l_{12}l_{34} - l_{13}l_{24} + l_{14}l_{23} = 0$$

- La matriz $\mathtt{L}$ es independiente de los puntos $\mathbf{A}$ y $\mathbf{B}$ elegidos; se llega a la misma matriz si se utilizan otros puntos alineados con aquellos.

Recta intersección de dos planos. Dados dos planos $\mathbf{P}$ y $\mathbf{Q}$, su recta intersección tiene la matriz dual $\mathtt{L}^* = \mathbf{P}\mathbf{Q}^\top - \mathbf{Q}\mathbf{P}^\top$, con propiedades similares a la matriz $\mathtt{L}$.

Los elementos de las matrices $\mathtt{L}^*$ y $\mathtt{L}$ guardan la siguiente relación de proporcionalidad:

$$(l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34}) \sim (l^*_{34}, l^*_{42}, l^*_{23}, l^*_{14}, l^*_{13}, l^*_{12})$$

Relaciones de incidencia Con esta representación matricial para las rectas, es fácil calcular el plano generado por un punto $\mathbf{X}$ y una recta $\mathtt{L}$, es $\boldsymbol{\pi} = \mathtt{L}^*\mathbf{X}$. Además, el punto $\mathbf{X}$ pertenece a la recta $\mathtt{L}$ sii $\mathtt{L}^*\mathbf{X} = \mathbf{0} \notin \mathbb{P}^3$.

La relación dual indica cómo obtener el punto de intersección de una recta $\mathtt{L}$ y un plano $\boldsymbol{\pi}$, es $\mathbf{X} = \mathtt{L}\boldsymbol{\pi}$. La recta está contenida en el plano sii $\mathtt{L}\boldsymbol{\pi} = \mathbf{0} \notin \mathbb{P}^3$.

Coordenadas de Plücker. Otra forma de representar las rectas es mediante los 6 elementos de la matriz antisimétrica explicada. El vector así creado recibe el nombre de coordenadas de Plücker de la recta. A su vez, hay varias posibilidades de definir el vector, las dos más habituales son:

$$\boldsymbol{\ell} \doteq (l_{12}, l_{13}, l_{14}, l_{23}, l_{24}, l_{34})^\top \quad \text{o} \quad \boldsymbol{\ell} \doteq (l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34})^\top \tag{3.3}$$

Obsérvese que lo único que cambia es el *quinto elemento* del vector. La segunda definición elimina el signo menos de la restricción $\det \mathtt{L} = 0$, ya que $l_{ij} = -l_{ji}$, es decir,

$$\det \mathtt{L} = 0 \Leftrightarrow l_{12}l_{34} + l_{13}l_{42} + l_{14}l_{23} = 0$$

Cualquiera de los dos vectores representa las coordenadas homogéneas de la recta y como son vectores de 6 componentes, pertenecen a $\mathbb{P}^5$. Todas las rectas de $\mathbb{P}^3$ son puntos de $\mathbb{P}^5$, sin embargo no es cierto lo inverso; para que un punto de $\mathbb{P}^5$ represente una recta de $\mathbb{P}^3$ sus coordenadas tienen que cumplir la restricción $\det \mathtt{L} = 0$. La interpretación geométrica de esta restricción es que las rectas de $\mathbb{P}^3$ forman una superficie cuádrica en $\mathbb{P}^5$, la llamada cuádrica de Klein. Es una cuádrica porque la relación

$\det \mathtt{L} = 0$ es cuadrática en los elementos del vector $\boldsymbol{\ell}$.

La matriz de la cuádrica de Klein tiene dos representaciones, según qué definición de (3.3) se elija:

$$\Omega = \begin{pmatrix} & & & & & 1 \\ & & & & -1 & \\ & & & 1 & & \\ & & 1 & & & \\ & -1 & & & & \\ 1 & & & & & \end{pmatrix} \quad \text{o} \quad \Omega = \begin{pmatrix} & & & & & 1 \\ & & & & 1 & \\ & & & 1 & & \\ & & 1 & & & \\ & 1 & & & & \\ 1 & & & & & \end{pmatrix}$$

Por ejemplo, para la segunda definición (3.3), la restricción $\det \mathtt{L} = 0$ se expresa en $\mathbb{P}^5$:

$$\boldsymbol{\ell}^\top \Omega \boldsymbol{\ell} = (l_{12}\ l_{13}\ l_{14}\ l_{23}\ l_{42}\ l_{34}) \begin{pmatrix} & & & & & 1 \\ & & & & 1 & \\ & & & 1 & & \\ & & 1 & & & \\ & 1 & & & & \\ 1 & & & & & \end{pmatrix} \begin{pmatrix} l_{12} \\ l_{13} \\ l_{14} \\ l_{23} \\ l_{42} \\ l_{34} \end{pmatrix} = 0 \tag{3.4}$$

Expresemos de forma concisa algunas propiedades importantes:

Pertenencia. Un vector $\boldsymbol{\ell} \in \mathbb{P}^5$ representa una recta de $\mathbb{P}^3$ sii pertenece a la cuádrica de Klein.

$$\boldsymbol{\ell}^\top \Omega \boldsymbol{\ell} = 0 \tag{3.5}$$

Incidencia Dos rectas $\boldsymbol{\ell}_1$, $\boldsymbol{\ell}_2$ de $\mathbb{P}^3$ son incidentes (se cortan) sii sus coordenadas de Plücker en $\mathbb{P}^5$ son conjugadas respecto de la cuádrica de Klein, Ω.

$$\boldsymbol{\ell}_1^\top \Omega \boldsymbol{\ell}_2 = 0 \tag{3.6}$$

Sólo cabe decir que las dos definiciones (3.3) se deben a que la primera es más cómoda de programar, aunque la segunda permite manejar las expresiones siempre con signos positivos.

3.3.3. Homografías

Acción de una homografía sobre puntos, planos, cuádricas y rectas

Si se aplica una homografía (transformación proyectiva representada por una matriz de 4×4 regular) a un punto: $\mathbf{X}' = \mathtt{H}\mathbf{X}$, entonces, para conservar la relación de incidencia de puntos y planos, éstos se transforman según la ecuación: $\boldsymbol{\pi}' = \mathtt{H}^{-\top}\boldsymbol{\pi}$, ya que $\boldsymbol{\pi}'^\top \mathbf{X}' = (\mathtt{H}^{-\top}\boldsymbol{\pi})^\top \mathtt{H}\mathbf{X} = \boldsymbol{\pi}^\top \mathtt{H}^{-1}\mathtt{H}\mathbf{X} = \boldsymbol{\pi}^\top \mathbf{X} = 0$.

Si el punto está sobre una cuádrica $\mathtt{Q}$, para que se verifique la ecuación de la cuádrica, ésta se transforma según $\mathtt{Q} \mapsto \mathtt{H}^{-\top}\mathtt{Q}\mathtt{H}^{-1}$. Si un plano está sobre una cuádrica dual $\mathtt{Q}^*$, para que se verifique la ecuación de pertenencia, la cuádrica se transforma de acuerdo a $\mathtt{Q}^* \mapsto \mathtt{H}\mathtt{Q}^*\mathtt{H}^\top$. Las demostraciones son similares a las dadas para las cónicas en § 3.2.7.

Según la homografía $\mathtt{H}$, la recta $\mathtt{L}$ se transforma en la recta $\mathtt{L}' = \mathtt{H}\mathtt{L}\mathtt{H}^\top$ y la recta $\mathtt{L}^*$ se transforma en la recta $\mathtt{L}^{*'} = \mathtt{H}^{-\top}\mathtt{L}^*\mathtt{H}^{-1}$.

Jerarquía de transformaciones

En $\mathbb{P}^3$ también hay una jerarquía de transformaciones. Una homografía arbitraria (proyectiva) está representada por una matriz regulara 4×4, por lo tanto posee $16 - 1 = 15$ grados de libertad. Pero existen subgrupos particulares, dentro del grupo de las transformaciones proyectivas.

El grupo de las afinidades o transformaciones afines, que posee 12 grados de libertad: una matriz regular arbitraria $\mathtt{A}$ y un vector de traslación $\mathbf{t}$. Lo más significativo de las transformaciones afines es que no varían las coordenadas del plano del infinito (queda fijo como conjunto, posiblemente no punto a punto).

$$\mathtt{H}_{\mathrm{A}} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{3.7}$$

El grupo conforme está formado por las transformaciones de semejanza, que tienen 7 grados de libertad. Una transformación de semejanza está formada por una rotación $\mathtt{R}$, un escalado s y una traslación $\mathbf{t}$.

$$\mathtt{H}_{\mathrm{S}} = \begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{3.8}$$

El último grupo interesante en la jerarquía es el grupo de las transformaciones euclídeas, que describe los movimientos de cuerpos rígidos, así que no hay dilatación ni contracción: $s = 1$. Por lo tanto quedan 6 grados de libertad: 3 para la rotación y 3 para la traslación. La matriz responde a la expresión:

$$\mathtt{H}_{\mathrm{E}} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{3.9}$$

3.4. Objetos de autocalibración

Presentamos los objetos geométricos que sirven para definir una estructura afín o una estructura euclídea en un espacio proyectivo. Por ejemplo, si tenemos una reconstrucción euclídea distorsionada, a la que llamamos reconstrucción proyectiva, y conocemos la posición de la cónica absoluta, podemos volver al mundo euclídeo.

3.4.1. El plano del infinito π_∞

- Es un plano especial dentro de los planos de $\mathbb{P}^3$.
- Expresión canónica (en un marco euclídeo): $W = 0 \Leftrightarrow \pi_\infty = (0:0:0:1)^\top$
- Es fijo ante transformaciones afines $\mathtt{H}_A$ (3.7). Es fijo como conjunto, no punto a punto. Esta propiedad es clave, ya que si se conocen las coordenadas del plano del infinito en una calibración proyectiva, es posible pasar a una calibración afín. Este es el primer paso de la calibración estratificada.
- El paralelismo se corresponde con compartir la incidencia (corte) en π_∞.
  - Dos rectas son paralelas si comparten el mismo punto de infinito.
  - Dos planos son paralelos si comparten la misma recta de infinito.
  - Una recta y un plano son paralelos si el punto de infinito de la recta está sobre la recta de infinito del plano.

3.4.2. La cónica absoluta Ω_∞

- Es una cónica de puntos contenida en π_∞. Está formada sólo por puntos imaginarios.
- Expresión canónica (en un marco euclídeo):

$$\Omega_\infty = \begin{cases} X^2 + Y^2 + Z^2 &= 0 \\ W &= 0 \end{cases}$$

  En π_∞, la matriz de la cónica es la identidad.

$$C = \mathtt{I} = \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} \Longleftrightarrow (X,Y,Z)\mathtt{I}(X,Y,Z)^\top = 0$$

- Es fija ante transformaciones de semejanza $\mathtt{H}_S$ (3.8). Es fija como conjunto, no punto a punto. Esta propiedad es también clave, pues si se conocen las ecuaciones de la cónica absoluta en una calibración proyectiva o una calibración afín, es posible pasar a una calibración métrica. Este es el segundo paso de la calibración estratificada.
- Cualquier recta de π_∞ corta a Ω_∞ en 2 puntos, como cumplen toda cónica y recta de un plano.
- Todas las circunferencias de $\mathbb{P}^3$ cortan a Ω_∞ en dos puntos: los puntos circulares del plano que contiene a la circunferencia.
- Todas las esferas cortan a π_∞ en Ω_∞.
- El ángulo entre dos direcciones $\mathbf{d}_1$ y $\mathbf{d}_2$ (puntos de π_∞) se puede determinar en cualquier marco proyectivo mediante la fórmula:

$$\cos(\theta) = \frac{\mathbf{d}_1^\top \Omega_\infty \mathbf{d}_2}{\sqrt{(\mathbf{d}_1^\top \Omega_\infty \mathbf{d}_1)(\mathbf{d}_2^\top \Omega_\infty \mathbf{d}_2)}} \tag{3.10}$$

  Caso particular: dos direcciones son ortogonales sii son conjugadas respecto de Ω_∞.

$$\theta = \pm\pi/2 \iff \mathbf{d}_1^\top \Omega_\infty \mathbf{d}_2 = 0 \tag{3.11}$$

3.4.3. La IAC y la DIAC

- La PAC o IAC es la proyección de la cónica absoluta (contenida en π_∞) sobre el plano de la imagen. Su expresión es $\omega = (\mathtt{K}\mathtt{K}^\top)^{-1}$
- La DIAC es la cónica de rectas dual de la proyección de la cónica absoluta. Su ecuación es $\omega^* = \mathtt{K}\mathtt{K}^\top$ y es la imagen de la cuádrica absoluta dual, $\mathtt{Q}_\infty^*$, mediante la fórmula: $\omega^* = \mathtt{P}\mathtt{Q}_\infty^*\mathtt{P}^\top$

Los puntos de π_∞ tienen coordenadas $\mathbf{X}_\infty = (\mathbf{d}^\top, 0)^\top$ y su proyección sobre el plano de la imagen es

$$\mathbf{x} \sim \mathtt{P}\mathbf{X}_\infty = \mathtt{K}\mathtt{R}[\mathtt{I} \mid -\widetilde{\mathbf{C}}] \begin{pmatrix} \mathbf{d} \\ 0 \end{pmatrix} = \mathtt{K}\mathtt{R}\mathbf{d}$$

La homografía que existe entre los puntos del plano del infinito y los puntos del plano de la imagen es $\mathtt{H} = \mathtt{K}\mathtt{R}$. Según esta homografía, la cónica absoluta Ω_∞ se transforma en $\omega = (\mathtt{K}\mathtt{K}^\top)^{-1} = \mathtt{K}^{-\top}\mathtt{K}^{-1}$, que recibe el nombre de proyección de la cónica absoluta (PAC) o imagen de la cónica absoluta (IAC).

Esto es fácil de comprobar: la matriz de la cónica absoluta en π_∞ es la identidad $\mathtt{C} = \mathtt{I}_3$, y las cónicas de puntos se transforman mediante homografías según $\mathtt{C} \mapsto \mathtt{H}^{-\top}\mathtt{C}\mathtt{H}^{-1}$. Sustituyendo en esta expresión la cónica y la homografía, $\mathtt{C} \equiv \Omega_\infty \mapsto (\mathtt{K}\mathtt{R})^{-\top}\mathtt{I}(\mathtt{K}\mathtt{R})^{-1} = \mathtt{K}^{-\top}\mathtt{R}\mathtt{R}^{-1}\mathtt{K}^{-1} = (\mathtt{K}\mathtt{K}^\top)^{-1} = \omega$.

La cónica dual recibe el nombre de DIAC (dual de la imagen de la cónica absoluta). Y como ya vimos en § 3.2.5, la matriz homogénea que la representa es

$$\omega^* = \omega^{-1} = \mathtt{K}\mathtt{K}^\top$$

Ambas cónicas se utilizan mucho en autocalibración, ya que sólo dependen de la matriz de parámetros intrínsecos y es la información que se desea recuperar para calibrar las cámaras. Tanto ω como ω^* determinan unívocamente $\mathtt{K}$ mediante la descomposición de Cholesky, que es aplicable ya que ambas matrices son definidas positivas (o negativas, dependiendo del factor de escala).

Cualquier plano π corta al plano del infinito π_∞ en una recta $\mathbf{l}_\infty$ y esta recta corta a Ω_∞ en dos puntos, que son los puntos circulares del plano π. La imagen de los puntos circulares está sobre ω, son los puntos en los que $\mathbf{l}_\infty$ corta a ω.

3.4.4. La cuádrica absoluta dual $\mathtt{Q}_\infty^*$

- La dual de Ω_∞ es $\mathtt{Q}_\infty^*$, una cuádrica dual degenerada, llamada la cuádrica absoluta dual. Cuádrica dual significa que es una cuádrica de planos, no de puntos. Es degenerada porque la matriz que la representa no es de rango máximo 4, sino 3.
- Expresión canónica (en un marco euclídeo):

$$\mathtt{Q}_\infty^* = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 0 \end{pmatrix}$$

- Al ser una cuádrica degenerada con un autovalor nulo, posee un núcleo no vacío, formado por el autovector asociado a dicho autovalor. π_∞ es el núcleo de $\mathtt{Q}_\infty^*$.

$$\mathtt{Q}_\infty^* \pi_\infty = 0 \tag{3.12}$$

- Es fija ante transformaciones de semejanza $\mathtt{H}_S$ (3.8). Es fija como conjunto, no punto a punto. La cuádrica absoluta dual contiene la información equivalente a la cónica absoluta y el plano del infinito, por lo que sirve para convertir una reconstrucción proyectiva en una reconstrucción métrica.
- El ángulo entre las normales de dos planos se puede determinar en cualquier marco proyectivo mediante la fórmula:

$$\cos(\theta) = \frac{\pi_1^\top \mathtt{Q}_\infty^* \pi_2}{\sqrt{(\pi_1^\top \mathtt{Q}_\infty^* \pi_1)(\pi_2^\top \mathtt{Q}_\infty^* \pi_2)}} \tag{3.13}$$

  Caso particular: dos planos son ortogonales sii son conjugados respecto de $\mathtt{Q}_\infty^*$.

$$\theta = \pm\pi/2 \iff \pi_1^\top \mathtt{Q}_\infty^* \pi_2 = 0 \tag{3.14}$$

3.4.5. La cuádrica degenerada Σ

Sea r una recta de $\mathbb{P}^3$ y $r = (l_{12}\ l_{13}\ l_{14}\ l_{23}\ l_{42}\ l_{34})^\top$ sus coordenadas de Plücker en $\mathbb{P}^5$. La restricción que verifican es

$$l_{12}l_{34} + l_{13}l_{24} + l_{14}l_{23} = 0$$

Dos rectas de $\mathbb{P}^3$ son ortogonales sii sus coordenadas de Plücker en $\mathbb{P}^5$ son conjugadas respecto de la cuádrica Σ, de rango 3.

$$r_1^\top \Sigma r_2 = 0, \qquad \Sigma = \begin{pmatrix} 0 & & & & & \\ & 0 & & & & \\ & & 1 & & & \\ & & & 0 & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix} \tag{3.15}$$

La cuádrica de Klein, Ω (§ 3.3) y la cuádrica de rango 3, Σ, son dos cuádricas de $\mathbb{P}^5$ que forman la base del haz de cuádricas que es el *Calibration Pencil* (ver [10]). Éste es también un objeto de autocalibración. La expresión canónica de Σ en un marco de referencia euclídeo es la matriz en (3.15).

3.5. Resumen

En la tabla 3.3 se recogen los conceptos más importantes de coordenadas homogéneas relativos a cónicas, cuádricas, sus duales y sus transformaciones según homografías.

Espacio	$\mathbb{P}^2$		$\mathbb{P}^3$	
Objeto	Cónica	Cónica Dual	Cuádrica	Cuádrica dual
Elemento	puntos $\mathbf{x}$	rectas $\mathbf{l}$	puntos $\mathbf{X}$	planos π
Ecuación	$\mathbf{x}^\top \mathtt{C}\mathbf{x} = 0$	$\mathbf{l}^\top \mathtt{C}^*\mathbf{l} = 0$	$\mathbf{X}^\top \mathtt{Q}\mathbf{X} = 0$	$\pi^\top \mathtt{Q}^*\pi = 0$
Polaridad	$\mathbf{l} = \mathtt{C}\mathbf{x}$	$\mathbf{x} = \mathtt{C}^*\mathbf{l}$	$\pi = \mathtt{Q}\mathbf{X}$	$\mathbf{X} = \mathtt{Q}^*\pi$
Dualidad	$\mathtt{C}^* \sim \mathtt{C}^{-1}$ si cónica no degenerada		$\mathtt{Q}^* \sim \mathtt{Q}^{-1}$ si cuádrica no degenerada	
Homografía	$\mathbf{x}' = \mathtt{T}\mathbf{x}$	$\mathbf{l}' = \mathtt{T}^{-\top}\mathbf{l}$	$\mathbf{X}' = \mathtt{H}\mathbf{X}$	$\pi' = \mathtt{H}^{-\top}\pi$
	$\mathtt{C}' = \mathtt{T}^{-\top}\mathtt{C}\mathtt{T}^{-1}$	$\mathtt{C}^{*\prime} = \mathtt{T}\mathtt{C}^*\mathtt{T}^\top$	$\mathtt{Q} = \mathtt{H}^{-\top}\mathtt{Q}\mathtt{H}^{-1}$	$\mathtt{Q}^{*\prime} = \mathtt{H}\mathtt{Q}^*\mathtt{H}^\top$

Cuadro 3.3: Resumen de las cónicas de $\mathbb{P}^2$ y cuádricas de $\mathbb{P}^3$

La tabla 3.4 resume la jerarquía de transformaciones en los grupos proyectivo (GP), afín (GA), conforme (GC) y euclídeo (SE), tanto en el plano proyectivo como en el espacio. También se detallan las formas de las matrices que representan las transformaciones y sus grados de libertad.

Espacio	$\mathbb{P}^2$				$\mathbb{P}^3$			
Grupo	GP(2)	GA(2)	GC(2)	SE(2)	GP(3)	GA(3)	GC(3)	SE(3)
Transformación	$\mathtt{T}_\mathrm{P}$ $\begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{v}^\top & k \end{bmatrix}$	$\mathtt{T}_\mathrm{A}$ $\begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$	$\mathtt{T}_\mathrm{S}$ $\begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$	$\mathtt{T}_\mathrm{E}$ $\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$	$\mathtt{H}_\mathrm{P}$ $\begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{v}^\top & k \end{bmatrix}$	$\mathtt{H}_\mathrm{A}$ $\begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$	$\mathtt{H}_\mathrm{S}$ $\begin{bmatrix} s\mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$	$\mathtt{H}_\mathrm{E}$ $\begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
Grados libertad	8	6	4	3	15	12	7	6
Notación	$\mathtt{A} \in \mathcal{M}_{2\times 2}$ regular; $\mathtt{R} \in SO(2)$; $\mathbf{t}, \mathbf{v} \in \mathbb{R}^2$; $k, s \in \mathbb{R}$				$\mathtt{A} \in \mathcal{M}_{3\times 3}$ regular; $\mathtt{R} \in SO(3)$; $\mathbf{t}, \mathbf{v} \in \mathbb{R}^3$; $k, s \in \mathbb{R}$			

Cuadro 3.4: Resumen de la jerarquía de transformaciones en $\mathbb{P}^2$ y $\mathbb{P}^3$

Capítulo 4

Estimación de homografías entre imágenes

En este capítulo se considera el problema de la estimación de transformaciones u otras entidades matemáticas basada en las medidas, observaciones de cierta naturaleza. Para ser más concretos, se dan las bases para estimar:

- **Homografías 2D**. Dadas unas correspondencias de puntos de $\mathbb{P}^2$, $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$, calcular la transformación proyectiva que lleva cada punto $\mathbf{x}_i$ a $\mathbf{x}'_i$. En una situación práctica, los puntos $\mathbf{x}_i$ y $\mathbf{x}'_i$ son puntos de dos imágenes, cada imagen considerada como un plano proyectivo $\mathbb{P}^2$.
- **Matrices de proyección**. Dado un conjunto de puntos $\mathbf{X}_i$ en el espacio y un conjunto correspondiente de puntos $\mathbf{x}_i$ en una imagen, calcular la proyección lineal que lleva los $\mathbf{X}_i$ sobre los $\mathbf{x}_i$. Tal proyección lineal es la operación realizada por una cámara proyectiva. A esta operación se le llama "resection" en la literatura anglosajona.
- **Matriz fundamental**. Dado un conjunto de puntos $\mathbf{x}_i$ en una imagen y los puntos correspondientes $\mathbf{x}'_i$ en la otra imagen, calcular una matriz fundamental $\mathtt{F}$ consistente.
- **Tensor trifocal**. Dadas unas correspondencias $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i \leftrightarrow \mathbf{x}''_i$ entre tres imágenes, calcular el tensor trifocal, el cual relaciona puntos y rectas en tres imágenes.

Estos problemas tienen muchas características comunes, así que la experiencia adquirida al resolver uno de ellos sirve para no cometer los mismos errores en la solución del resto. En este capítulo se considera el primero de los problemas.

4.1. Introducción

Una aplicación directa del cálculo de homografías entre pares de imágenes surge al utilizar una cámara rotatoria para adquirir las mismas. El vector de traslación entre las posiciones de las cámaras es nulo. Bajo esta hipótesis, los puntos proyectados en las imágenes se relacionan mediante una homografía del plano y es posible obtener la matriz de parámetros intrínsecos de la cámara en cada imagen a partir de homografías. Otra aplicación es fotos de fotos: la homografía describe la relación punto a punto que existe entre los puntos de un plano (una foto) y las proyecciones de éstos en otro plano del espacio.

El enunciado del problema es el siguiente: Dadas $n \geq 4$ parejas de puntos en cada imagen $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determina la homografía de $\mathbb{P}^2$ cuya matriz es $\mathtt{H}$ (3×3) tal que $\mathbf{x}'_i \sim \mathtt{H}\mathbf{x}_i$, $\forall i$.

Son necesarias, al menos, 4 parejas de puntos para calcular la matriz de la transformación. Explicación: la matriz $\mathtt{H}$ tiene 9 elementos, pero está definida a falta de un factor de proporcionalidad; así

que tiene $9-1$ grados de libertad. Por otro lado, cada pareja de puntos proporciona dos restricciones sobre cómo se transforman 2 de sus coordenadas; también están definidos salvo proporcionalidad.

La solución $\mathtt{H}$ es única si sólo se proporcionan 4 parejas de puntos. En presencia de ruido y más de 4 correspondencias de puntos las parejas de puntos pueden no ser compatibles con la misma transformación proyectiva, y en estos casos es deseable estimar la "mejor" transformación $\mathtt{H}$ que se ajusta a los datos. El sentido "mejor" se define relativo a cierta función de coste a minimizar. Hay dos grandes familias de funciones de coste: las basadas en minimizar un error algebraico (cómo se verifican las ecuaciones) y las que minimizan un error geométrico: una cantidad medible en las imágenes.

El término "Gold Standard" se utiliza a lo largo de la memoria para referirse al algoritmo que minimiza la función de coste que proporciona la mejor estimación $\mathtt{H}$ (es óptima) bajo ciertas hipótesis: el ruido está confinado a las coordenadas de los puntos observados en las imágenes, sigue una distribución gaussiana de media nula y es independiente en las dos coordenadas x, y de cada punto.

4.2. Algoritmo lineal

En general, los algoritmos lineales se basan en la teoría de mínimos cuadrados, propuesta por Gauss. Para resolver el problema se utilizan técnicas estándar de álgebra lineal matricial: principalmente la descomposición en autovalores y autovectores, junto con la descomposición en valores singulares (apéndice B). A veces se utilizan las siglas DLT (Direct Linear Transformation) o SVD para referirse a los algoritmos lineales.

Se utilizan como inicialización de los algoritmos no lineales. Las funciones de coste que minimizan son de la familia de las algebraicas: minimizan, habitualmente sin imponer restricciones, el grado de verificación de una ecuación implícita, definido como veremos más adelante.

El algoritmo lineal para el cálculo de homografías del plano se puede consultar en [1, pág. 73]. La expresión $\mathbf{x}'_i \sim \mathtt{H}\mathbf{x}_i$ indica que los vectores a cada lado del símbolo $\sim$ son proporcionales. En términos del producto vectorial se expresa lo mismo, pero en forma de ecuaciones implícitas: $\mathbf{x}'_i \times \mathtt{H}\mathbf{x}_i = \mathbf{0}$. Esta última fórmula son tres ecuaciones lineales en las entradas de la matriz de la homografía $\mathtt{H}$, de las cuales sólo dos son linealmente independientes. Denotando $\mathbf{x}_i = (x_i, y_i, w_i)^\top$, $\mathbf{x}'_i = (x'_i, y'_i, w'_i)^\top$ y $\mathbf{h}^{j\top}$ la j-ésima fila de la matriz $\mathtt{H}$, las 3 ecuaciones son, en forma de sistema:

$$\mathtt{A}_i\mathbf{h} = \begin{bmatrix} \mathbf{0}^\top & -w'_i\mathbf{x}_i^\top & y'_i\mathbf{x}_i^\top \\ w'_i\mathbf{x}_i^\top & \mathbf{0}^\top & -x'_i\mathbf{x}_i^\top \\ -y'_i\mathbf{x}_i^\top & x'_i\mathbf{x}_i^\top & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0}$$

Esta ecuación es lineal en los elementos de $\mathbf{h}$. Los elementos de la matriz $\mathtt{A}_i$ son cuadráticos en las coordenadas de los puntos observados. Denominaremos "matriz de diseño de la pareja de puntos" a la matriz $\mathtt{A}_i$, que, dicho sea de paso, es de tamaño 3×9. Concatenando matrices de diseño de puntos se forma la matriz de diseño del problema. Con $n = 4$ parejas de puntos creamos una matriz $\mathtt{A}$ de $3n \times 9 = 12 \times 9$,

$$\mathtt{A} = \begin{bmatrix} \mathtt{A}_1 \\ \mathtt{A}_2 \\ \vdots \\ \mathtt{A}_n \end{bmatrix}$$

Esta matriz es de rango 8 (si los puntos están en posición general), por lo que el sistema homogéneo $\mathtt{A}\mathbf{h} = \mathbf{0}$ tiene solución no trivial exacta y única (salvo factor de proporcionalidad). En cambio, si se utilizan $n > 4$ puntos, puede que la matriz sea de rango 9 y ya no tenga solución exacta (salvo la trivial). En tal situación se busca una solución mínimo-cuadrática.

En presencia de ruido se puede enunciar el siguiente problema de optimización:

$$\min_{\mathbf{h}} \|\mathtt{A}\mathbf{h}\| \quad \text{sujeto a} \quad \|\mathbf{h}\| = 1$$

Este problema es un clásico al haber impuesto, sin pérdida de generalidad, $\|\mathbf{h}\| = 1$, ya que el factor de proporcionalidad no afecta a la solución. La función de coste $\|\mathtt{A}\mathbf{h}\|$ se conoce como distancia algebraica.

La solución se explica en § B.3.2 y se puede hallar por dos procedimientos, básicamente: mediante la descomposición en valores singulares de la matriz de diseño $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^\top$ o mediante los autovalores y autovectores de $\mathtt{A}^\top\mathtt{A} = \mathtt{V}\mathtt{D}^2\mathtt{V}^{-1}$.

> **Solución SVD**: $\mathbf{h}$ es la columna de $\mathtt{V}$ correspondiente al menor valor singular de $\mathtt{A}$, es decir, la novena columna.
>
> **Solución EIG**: $\mathbf{h}$ es la columna de $\mathtt{V}$ correspondiente al menor autovalor de $\mathtt{A}^\top\mathtt{A}$.

Existe la discusión general de estos dos caminos: la matriz $\mathtt{A}^\top\mathtt{A}$ siempre es una matriz cuadrada del tamaño del número de incógnitas: 9×9, sea cual sea el número de puntos $n \geq 4$ utilizados en la estimación, en cambio la matriz $\mathtt{A}$ es de $3n \times 9$, crece con el número de puntos.

Otra consideración es la estabilidad numérica del sistema debido a la precisión finita: para que el algoritmo sea útil, la diferencia de órdenes de magnitud del mayor y el segundo menor valor singular/autovalor no debe ser muy grande. Esta medida no es exactamente el número de condición de una matriz, ya que éste se define entre los mayor y menor valores singulares/autovalores. Para disminuir esta diferencia de órdenes de magnitud es muy popular realizar una transformación afín previa de las coordenadas de los puntos observados en las imágenes, según [5]. Por ahora posponemos los detalles de esa normalización afín, aunque es lo que realmente hace que el algoritmo lineal sea práctico.

También es importante la complejidad de las descomposiciones (número de operaciones): lo que se ahorra en una descomposición sencilla de autovectores y autovalores (debido a las pequeñas dimensiones de la matriz de $n \times n$) se gasta en realizar el producto matricial $\mathtt{A}^\top\mathtt{A}$; en cambio, en la descomposición en valores singulares no hay necesidad de realizar el producto matricial (ahorro), pero la matriz de diseño $\mathtt{A}$ crece con el número de puntos y también lo hace la complejidad de la descomposición (ver § B.4).

Coste del ajuste

El coste es el menor valor singular de la matriz de diseño $\mathtt{A}$, es decir, el mínimo de la distancia algebraica $\sigma_9 = \min \|\mathtt{A}\mathbf{h}\| = \min \|\epsilon\|$. Este coste resulta útil para comparar ajustes con otros algoritmos que manejen la misma distancia.

Otra posible función de coste es aquella para la que el coste es el cociente σ_9/σ_1 en el caso de valores singulares ($\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_9$) o $\sqrt{\lambda_9/\lambda_1}$ en el caso de autovalores ($\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_9$). Tales cocientes son iguales y dan una idea de la singularidad de la matriz $\mathtt{A}$ independientemente de su escala. Sin embargo, preferimos la primera medida, por tener una relación directa con la norma del vector de error o residuo, $\epsilon = \mathtt{A}\mathbf{h}$.

Variaciones sobre el algoritmo lineal

Una forma más eficiente de resolver el sistema $\mathtt{A}\mathbf{h} = \mathbf{0}$ es reunir todos los ceros de $\mathtt{A}$ que se pueda, de manera que resulte una matriz levemente dispersa. Matemáticamente, el sistema tiene la misma solución independientemente del orden de las ecuaciones. Para el caso de n parejas de puntos:

$$\mathtt{A} = \begin{bmatrix} \mathbf{0}^\top & -w'_1\mathbf{x}_1^\top & y'_1\mathbf{x}_1^\top \\ w'_1\mathbf{x}_1^\top & \mathbf{0}^\top & -x'_1\mathbf{x}_1^\top \\ -y'_1\mathbf{x}_1^\top & x'_1\mathbf{x}_1^\top & \mathbf{0}^\top \\ \mathbf{0}^\top & -w'_2\mathbf{x}_2^\top & y'_2\mathbf{x}_2^\top \\ w'_2\mathbf{x}_2^\top & \mathbf{0}^\top & -x'_2\mathbf{x}_2^\top \\ -y'_2\mathbf{x}_2^\top & x'_2\mathbf{x}_2^\top & \mathbf{0}^\top \\ \vdots & \vdots & \vdots \\ \mathbf{0}^\top & -w'_n\mathbf{x}_n^\top & y'_n\mathbf{x}_n^\top \\ w'_n\mathbf{x}_n^\top & \mathbf{0}^\top & -x'_n\mathbf{x}_n^\top \\ -y'_n\mathbf{x}_n^\top & x'_n\mathbf{x}_n^\top & \mathbf{0}^\top \end{bmatrix} \equiv \begin{bmatrix} \mathbf{0}^\top & -w'_1\mathbf{x}_1^\top & y'_1\mathbf{x}_1^\top \\ \mathbf{0}^\top & -w'_2\mathbf{x}_2^\top & y'_2\mathbf{x}_2^\top \\ \vdots & \vdots & \vdots \\ \mathbf{0}^\top & -w'_n\mathbf{x}_n^\top & y'_n\mathbf{x}_n^\top \\ w'_1\mathbf{x}_1^\top & \mathbf{0}^\top & -x'_1\mathbf{x}_1^\top \\ w'_2\mathbf{x}_2^\top & \mathbf{0}^\top & -x'_2\mathbf{x}_2^\top \\ \vdots & \vdots & \vdots \\ w'_n\mathbf{x}_n^\top & \mathbf{0}^\top & -x'_n\mathbf{x}_n^\top \\ -y'_1\mathbf{x}_1^\top & x'_1\mathbf{x}_1^\top & \mathbf{0}^\top \\ -y'_2\mathbf{x}_2^\top & x'_2\mathbf{x}_2^\top & \mathbf{0}^\top \\ \vdots & \vdots & \vdots \\ -y'_n\mathbf{x}_n^\top & x'_n\mathbf{x}_n^\top & \mathbf{0}^\top \end{bmatrix} \equiv \begin{bmatrix} \mathbf{0} & -\widetilde{\mathtt{A}}_3 & \widetilde{\mathtt{A}}_2 \\ \widetilde{\mathtt{A}}_3 & \mathbf{0} & -\widetilde{\mathtt{A}}_1 \\ -\widetilde{\mathtt{A}}_2 & \widetilde{\mathtt{A}}_1 & \mathbf{0} \end{bmatrix}$$

El sistema, al ponerlo en forma de bloques, queda:

$$\begin{bmatrix} \mathbf{0} & -\widetilde{\mathtt{A}}_3 & \widetilde{\mathtt{A}}_2 \\ \widetilde{\mathtt{A}}_3 & \mathbf{0} & -\widetilde{\mathtt{A}}_1 \\ -\widetilde{\mathtt{A}}_2 & \widehat{\mathtt{A}}_1 & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0} \equiv \begin{cases} -\widetilde{\mathtt{A}}_3\mathbf{h}^2 + \widetilde{\mathtt{A}}_2\mathbf{h}^3 &= 0 \\ \widetilde{\mathtt{A}}_3\mathbf{h}^1 - \widetilde{\mathtt{A}}_1\mathbf{h}^3 &= 0 \\ -\widetilde{\mathtt{A}}_2\mathbf{h}^1 + \widetilde{\mathtt{A}}_1\mathbf{h}^2 &= 0 \end{cases}$$

En las dos primeras ecuaciones podemos despejar $\mathbf{h}^1$ y $\mathbf{h}^2$ en función de $\mathbf{h}^3$ y resolver el sistema para este último. Llamamos $\mathtt{A}^+$ a la pseudo-inversa de $\mathtt{A}$. Para más aclaraciones se recomienda consultar los apéndices.

$$\left.\begin{array}{rcl} \mathbf{h}^2 &=& \widetilde{\mathtt{A}}_3^+\widetilde{\mathtt{A}}_2\mathbf{h}^3 \\ \mathbf{h}^1 &=& \widetilde{\mathtt{A}}_3^+\widetilde{\mathtt{A}}_1\mathbf{h}^3 \\ 0 &=& \widetilde{\mathtt{A}}_2\mathbf{h}^1 - \widetilde{\mathtt{A}}_1\mathbf{h}^2 \end{array}\right\} \Rightarrow \widetilde{\mathtt{A}}_2\widetilde{\mathtt{A}}_3^+\widetilde{\mathtt{A}}_1\mathbf{h}^3 - \widetilde{\mathtt{A}}_1\widetilde{\mathtt{A}}_3^+\widetilde{\mathtt{A}}_2\mathbf{h}^3 = (\widetilde{\mathtt{A}}_2\widetilde{\mathtt{A}}_3^+\widetilde{\mathtt{A}}_1 - \widetilde{\mathtt{A}}_1\widetilde{\mathtt{A}}_3^+\widetilde{\mathtt{A}}_2)\mathbf{h}^3 = \mathtt{M}\mathbf{h}^3 = 0$$

El sistema homogéneo inicial $\mathtt{A}\mathbf{h} = \mathbf{0}$, cuya matriz de diseño era de $3n \times 9$, queda reducido a un sistema homogéneo $\mathtt{M}\mathbf{h}^3 = 0$, cuya matriz de diseño es de $n \times 3$. El algoritmo, en lugar de hallar la SVD de la matriz $\mathtt{A}$, calcula la solución mediante la pseudoinversa de una matriz $\widetilde{\mathtt{A}}_3$ de tamaño $n \times 3$ y la SVD de la nueva matriz de diseño, $\mathtt{M}$.

Comparación de la complejidad computacional

El número aproximado de operaciones en coma flotante necesario para calcular la descomposición SVD de una matriz de dimensiones $F \times C$ es un total de $4F^2C + 8FC^2 + 9C^3$, según § B.4. Sin embargo, si sólo se necesitan las matrices $\mathtt{D}$ y $\mathtt{V}$, el número de operaciones es $4FC^2 + 8C^3$. La distinción es importante puesto que en la última fórmula no hay un término dependiente de F^2. Recordemos que en los sistemas sobredeterminados (más ecuaciones que incógnitas) hay normalmente muchas más filas que columnas en la matriz de diseño. Es recomendable no calcular la matriz $\mathtt{U}$ a menos que sea indispensable.

El paso costoso del algoritmo lineal original es la SVD de $\mathtt{A}$ ($3n \times 9$), mientras que en el algoritmo modificado podemos considerar que los pasos costosos son calcular $\widetilde{\mathtt{A}}_3^+$ y la SVD de $\mathtt{M}$, aproximadamente, dos SVD's ($n \times 3$).

	Algoritmo lineal	Variación del algoritmo lineal
SVD ($\mathtt{U}$, $\mathtt{D}$, $\mathtt{V}$) $4F^2C + 8FC^2 + 9C^3$	$4(3n)^2 9 + 8(3n)9^2 + 9(9)^3$ $324n^2 + 1944n + 6561$	$2(4n^2 3 + 8n3^2 + 9(3)^3)$ $24n^2 + 144n + 486$
SVD ($\mathtt{D}$, $\mathtt{V}$) $4FC^2 + 8C^3$	$4(3n)9^2 + 8(9)^3$ $972n + 5832$	$2(4n3^2 + 8(3)^3)$ $72n + 432$

Para un número elevado de parejas de puntos, la mejora en rapidez para la descomposición SVD completa es:

$$\lim_{n\to\infty} \frac{324n^2 + 1944n + 6561}{24n^2 + 144n + 486} = \frac{324}{24} = 13{,}5$$

Es decir, el algoritmo modificado es 13,5 veces más rápido que el algoritmo original, así que puede resultar provechoso si se utiliza para proporcionar un punto de partida para un algoritmo iterativo.

En toda esta discusión se supone que ambos algoritmos proporcionan soluciones de la misma magnitud en cuanto a distancia algebraica: $\|\mathbf{Ah}\|$. Esto es así en el caso de poco ruido en los datos. En las pruebas experimentales se evalúa esta modificación del algoritmo.

4.3. Algoritmos Iterativos

Los algoritmos no lineales persiguen la minimización de una función de coste o distancia mediante esquemas iterativos, habitualmente fundamentados en la suposición de que la función a minimizar es localmente lineal.

Estos algoritmos son muy útiles porque permiten minimizar funciones de coste de ambas familias de error: algebraico y geométrico, el último es medible en las imágenes. A la vez que minimizan un coste, suelen imponer restricciones propias que debe satisfacer la geometría de la solución, mediante una parametrización adecuada del espacio de soluciones.

Las funciones de coste de ambas familias se pueden estudiar bajo un marco descriptivo común. Es muy importante entender esta formulación de los problemas, ya que se usa muchísimo a lo largo de toda el proyecto. El marco consta de los siguientes elementos:

1. un vector de parámetros $\mathbf{P} \in \mathbb{R}^M$ (variables independientes respecto a las cuales minimizar)
2. un vector de medidas $\mathbf{X} \in \mathbb{R}^N$ (variables dependientes) con matriz de covarianzas $\Sigma_{\mathbf{X}}$.
3. una función (modelo) $f : \mathbb{R}^M \to \mathbb{R}^N$, cuyo recorrido es, localmente, el conjunto de medidas permitidas.
4. la función de coste a minimizar se expresa como el cuadrado de la distancia de Mahalanobis:

$$\|\mathbf{X} - f(\mathbf{P})\|^2_{\Sigma_{\mathbf{X}}} = (\mathbf{X} - f(\mathbf{P}))^\top \Sigma_{\mathbf{X}}^{-1} (\mathbf{X} - f(\mathbf{P})) \tag{4.1}$$

Esta descripción permite la aplicación directa del algoritmo de Levenberg-Marquardt (LM), descrito en § D.1.1. También es recomendable una lectura del apéndice 4 de [1, pág. 568].

En la mayoría de las ocasiones, se supone que las medidas son independientes. Ninguna pareja de puntos u observación es más importante que otra, por lo que la matriz de covarianzas $\Sigma_{\mathbf{X}}$ es la identidad. Sin embargo, se podría cambiar la importancia de cada pareja de puntos para implementar un algoritmo de estimación robusto, similar al RANSAC § 4.4, dando más peso a unas parejas (similar a los *inliers* de RANSAC), que a otras (*outliers*). Se implementa así una versión de mínimos cuadrados ponderados.

Durante la ejecución del proyecto se han implementado las tres funciones de coste geométrico del capítulo 3 de [1], aunque en la práctica siempre se utilice la óptima (Gold Standard), consistente en minimizar el error de reproyección en las dos imágenes.

Utilizaremos la siguiente notación: $\mathbf{x}$ representa coordenadas de los puntos medidos (observados); $\hat{\mathbf{x}}$ representa valores estimados de los puntos y $\bar{\mathbf{x}}$ representa valores exactos de los puntos.

4.3.1. Error en una imagen

La función de coste geométrico más sencilla es la que sólo considera que hay errores en una de las dos imágenes, supongamos que es la segunda (la que denotamos con primas). Esto no es verdad en la mayoría de las aplicaciones prácticas con imágenes. En en cambio sí encaja en un problema en el que se desee ajustar una imagen a un patrón. La cantidad a minimizar es el error de transferencia de los puntos de la primera imagen a la segunda según la homografía, es decir, el error en la segunda imagen entre los puntos observados $\mathbf{x}'$ y los puntos transferidos $\mathtt{H}\bar{\mathbf{x}}_i$.

$$\sum_i d(\mathbf{x}'_i, \mathtt{H}\bar{\mathbf{x}}_i)^2$$

donde $d(\mathbf{x}, \mathbf{y})$ es la distancia euclídea entre las coordenadas deshomogeneizadas de los puntos $\mathbf{x}$ e $\mathbf{y}$.

La función modelo en el marco descriptivo común introducido al principio de esta sección es:

$$f : \mathbf{h} \equiv \mathbf{P} \rightarrow \widehat{\mathbf{X}} \equiv (\mathtt{H}\mathbf{x}_1, \ldots, \mathtt{H}\mathbf{x}_n)$$

El vector de parámetros $\mathbf{P}$ de $M = 9$ componentes indica que sólo se optimiza respecto de la homografía. El vector de medidas $\widehat{\mathbf{X}}$, de dimensión $N = 2n$, se forma con las coordenadas afines de los puntos transferidos, no con las coordenadas homogéneas como pueda parecer en la notación: $\mathtt{H}\bar{\mathbf{x}}_i$. La función de coste expresada en términos de la función modelo es, suponiendo que todas las medidas tienen la misma importancia:

$$\text{coste} = \|\mathbf{X} - \widehat{\mathbf{X}}\| = \|\mathbf{X} - f(\mathbf{P})\| = \sqrt{\sum_i d(\mathbf{x}'_i, \mathtt{H}\mathbf{x}_i)^2}$$

Más referencias sobre este algoritmo en [1, pág. 77, 86 y 96].

4.3.2. Error de Transferencia Simétrico

En casos más reales, los errores en la determinación de las posiciones de los puntos se producen en ambas imágenes y es preferible minimizar el error en ambas imágenes. Una forma de hacerlo es considerar las transformaciones directa $\mathtt{H}$ e inversa $\mathtt{H}^{-1}$. Se transfieren los puntos de la primera imagen a la segunda según $\mathtt{H}$; se transfieren los puntos de la segunda imagen a la primera mediante $\mathtt{H}^{-1}$, y se suman las distancias geométricas a los respectivos puntos:

$$\sum_i \left[d(\mathbf{x}_i, \mathtt{H}^{-1}\mathbf{x}'_i)^2 + d(\mathbf{x}'_i, \mathtt{H}\mathbf{x}_i)^2 \right]$$

La función modelo en este caso es:

$$f : \mathbf{h} \equiv \mathbf{P} \rightarrow \widehat{\mathbf{X}} \equiv (\mathtt{H}^{-1}\mathbf{x}'_1, \ldots, \mathtt{H}^{-1}\mathbf{x}'_n, \mathtt{H}\mathbf{x}_1, \ldots, \mathtt{H}\mathbf{x}_n)$$

El vector de parámetros $\mathbf{P}$ sigue teniendo $M = 9$ componentes, pero el vector de medidas duplica su tamaño: $N = 4n$. La función de coste en términos de la función modelo sigue la ecuación (4.1). Para más detalles, consultar [1, pág. 78 y 96] .

$$\text{coste} = \|\mathbf{X} - \widehat{\mathbf{X}}\| = \|\mathbf{X} - f(\mathbf{P})\| = \sqrt{\sum_i \left[d(\mathbf{x}_i, \mathtt{H}^{-1}\mathbf{x}'_i)^2 + d(\mathbf{x}'_i, \mathtt{H}\mathbf{x}_i)^2 \right]}$$

4.3.3. Error de Reproyección

El algoritmo óptimo en el sentido de buscar una estimación de máxima verosimilitud de $\mathtt{H}$ bajo la hipótesis de ruido habitual es el que minimiza la suma de errores en las dos imágenes a la vez

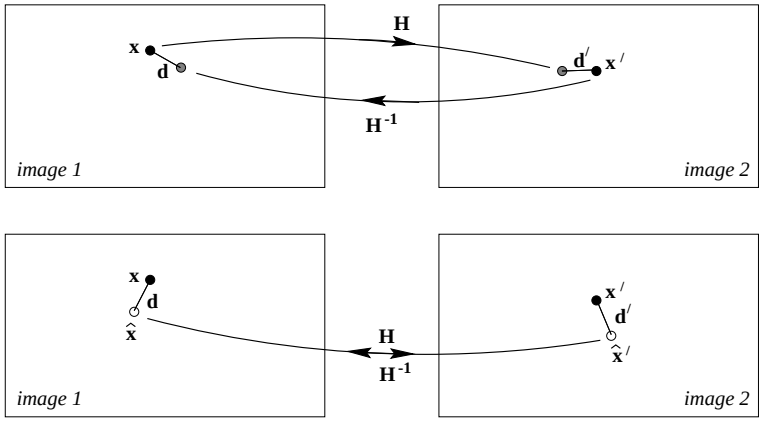


Figura 4.1: Comparación conceptual entre el error de transferencia simétrico (arriba) y el error de reproyección (abajo) al estimar una homografía.

que obtiene unas "correcciones" de los puntos en las imágenes, de tal forma que son una verdadera correspondencia de puntos según la homografía. Se busca la homografía $\mathtt{H}$ y parejas de puntos $\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i$ que minimizan

$$\sum_i \left[d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \right] \quad \text{sujeto a} \quad \hat{\mathbf{x}}'_i = \mathtt{H}\hat{\mathbf{x}}_i \; \forall i$$

Este problema de optimización requiere determinar tanto $\mathtt{H}$ como un conjunto de correspondencias $\{\hat{\mathbf{x}}_i\}$ y $\{\hat{\mathbf{x}}'_i\}$.

Analicemos el diseño del problema desde el punto de vista del marco común del algoritmo LM. La función modelo es:

$$f : (\mathbf{h}, \hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n) \equiv \mathbf{P} \rightarrow \widehat{\mathbf{X}} \equiv (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}'_1, \ldots, \hat{\mathbf{x}}_n, \hat{\mathbf{x}}'_n)$$

sujeto a $\hat{\mathbf{x}}'_i = \mathtt{H}\hat{\mathbf{x}}_i$, es decir, los puntos corregidos verifican la ecuación de la homografía.

La dimensión del vector de parámetros $\mathbf{P}$ es $M = 2n + 9$ componentes (considerablemente mayor que en los dos algoritmos anteriores). El vector de medidas es de dimensión $N = 4n$, como el del error de transferencia simétrico. La función de coste según la ec. (4.1) es

$$\text{coste} = \|\mathbf{X} - \widehat{\mathbf{X}}\| = \|\mathbf{X} - f(\mathbf{P})\| = \sqrt{\sum_i \left[d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \right]}$$

¿Por qué recibe esta función de coste el nombre de error de reproyección? Porque este problema de estimación modela la situación en la que los puntos $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ son proyecciones de un mismo punto en un plano del espacio, según ilustra la figura 4.2. Queremos estimar el punto del plano del espacio $\hat{\mathbf{X}}_i$ a partir de $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ para *reproyectarlo* y obtener la verdadera correspondencia $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ que mejor se ajusta. Más referencias: [1, pág. 78, 85-87 y 96].

Inicialización

Como vector de parámetros inicial de la búsqueda mediante los distintos algoritmos de optimización no lineal se utiliza la homografía estimada gracias a la solución del algoritmo lineal. La inicialización es un paso muy importante para encontrar el mínimo coste deseado: si conseguimos poner el punto inicial de la búsqueda en la zona de atracción del mínimo, los algoritmos deterministas de optimización funcionan muy bien. Esto se volverá a subrayar más adelante.

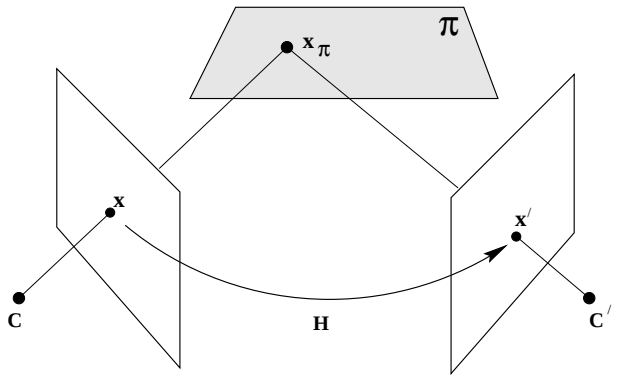


Figura 4.2: Homografía inducida por un plano

Iteración

Para minimizar cualquiera de los tres costes geométricos definidos anteriormente se necesitan algoritmos de optimización no lineal. La opción recomendada es utilizar una implementación del algoritmo de Levenberg-Marquardt que esté adaptada al problema. Las distintas variaciones del algoritmo se consideran rutinas de propósito general y están descritas en otro apartado. Sin embargo, comentaremos aquí la optimización adaptada al algoritmo Gold Standard de cálculo de homografías.

Antes de proseguir es recomendable una lectura del apéndice 4 de [1], en especial de los apartados A.4.4, A.4.4.1., junto con el capítulo del presente documento dedicado a las rutinas Levenberg-Marquardt, para entender qué se está describiendo.

Como el vector de parámetros está dividido en dos partes (los parámetros de la homografía y los parámetros de los puntos corregidos) la matriz jacobiana de la función modelo presenta una estructura de bloques: los bloques que representan las derivadas respecto de la homografía son las submatrices $\mathtt{A}_i$ y los bloques que representan las derivadas respecto de los parámetros de los puntos corregidos son las submatrices $\mathtt{B}_i$.

La forma de estas submatrices jacobianas $\mathtt{A}_i(4\times 9)$ y $\mathtt{B}_i(4\times 2)$ se conoce analíticamente y se pueden construir a partir de los vectores $\mathbf{P}$ y $f(\mathbf{P})$ en cualquier paso del algoritmo, por lo que no hace falta estimar la matriz jacobiana numéricamente. Suponiendo que un punto $\hat{\mathbf{x}}_i$ tiene por coordenadas homogéneas $\hat{\mathbf{x}}_{ih} = (x_i, y_i, w_i)^\top$, y el punto $\hat{\mathbf{x}}'_i$ tiene coordenadas homogéneas $\hat{\mathbf{x}}'_{ih} = \mathtt{H}\hat{\mathbf{x}}_{ih} = (x'_i, y'_i, w'_i)^\top$, las coordenadas afines que componen $\widehat{\mathbf{X}}$ son:

$$\widehat{\mathbf{X}}_i = \begin{pmatrix} \hat{\mathbf{x}}_i \\ \hat{\mathbf{x}}'_i \end{pmatrix} \quad ,, \quad \hat{\mathbf{x}}_i = \left(\frac{x_i}{w_i}, \frac{y_i}{w_i}\right)^\top \quad ,, \quad \hat{\mathbf{x}}'_i = \left(\frac{x'_i}{w'_i}, \frac{y'_i}{w'_i}\right)^\top$$

Las matrices jacobianas son (derivadas exactas):

$$\mathtt{A}_i = \frac{\partial \widehat{\mathbf{X}}_i}{\partial \mathbf{h}} = \begin{bmatrix} \mathbf{0} \\ \dfrac{\partial \hat{\mathbf{x}}'_i}{\partial \mathbf{h}} \end{bmatrix} \quad ,, \quad \frac{\partial \hat{\mathbf{x}}'_i}{\partial \mathbf{h}} = \frac{1}{w'_i} \begin{bmatrix} \tilde{\mathbf{x}}_i^\top & \mathbf{0}^\top & -\dfrac{x'_i}{w'_i}\tilde{\mathbf{x}}_i^\top \\ \mathbf{0}^\top & \tilde{\mathbf{x}}_i^\top & -\dfrac{y'_i}{w'_i}\tilde{\mathbf{x}}_i^\top \end{bmatrix} = \begin{bmatrix} \mathtt{I} & -\hat{\mathbf{x}}'_i \end{bmatrix} \otimes \frac{1}{w'_i}\tilde{\mathbf{x}}_i^\top$$

siendo $\tilde{\mathbf{x}}_i^\top = (x_i, y_i, 1)$

$$\mathtt{B}_i = \frac{\partial \widehat{\mathbf{X}}_i}{\partial \hat{\mathbf{x}}_i} = \begin{bmatrix} \mathtt{I} \\ \frac{\partial \hat{\mathbf{x}}_i'}{\partial \hat{\mathbf{x}}_i} \end{bmatrix} \quad , , \quad \frac{\partial \hat{\mathbf{x}}_i'}{\partial \hat{\mathbf{x}}_i} = \frac{1}{w_i'} \begin{bmatrix} \mathtt{H}_{11} - \frac{x_i'}{w_i'}\mathtt{H}_{31} & \mathtt{H}_{12} - \frac{x_i'}{w_i'}\mathtt{H}_{32} \\ \mathtt{H}_{21} - \frac{y_i'}{w_i'}\mathtt{H}_{31} & \mathtt{H}_{22} - \frac{y_i'}{w_i'}\mathtt{H}_{32} \end{bmatrix} = \frac{1}{w_i'} \left(\begin{bmatrix} \mathtt{H}_{11} & \mathtt{H}_{12} \\ \mathtt{H}_{21} & \mathtt{H}_{22} \end{bmatrix} - \hat{\mathbf{x}}_i' [\mathtt{H}_{31}\ \mathtt{H}_{32}] \right)$$

4.4. Estimación robusta de homografías mediante RANSAC

También se ha implementado un algoritmo para calcular la homografía a partir de correspondencias de puntos entre dos imágenes, suponiendo que no todas las parejas son correctas. La estimación robusta mediante RANSAC (RANdom SAmple Consensus) consiste en la estimación de la homografía y además un conjunto de parejas de puntos consistentes con dicha estimación (las verdaderas correspondencias, llamadas *inliers*), junto con las correspondencias que no cumplen el modelo, los *outliers*. La referencia que se ha seguido es [1, pág. 101].

Hasta ahora se había supuesto que la única fuente de error en las correspondencias $\{\mathbf{x}_i \leftrightarrow \mathbf{x}_i'\}$ era error aditivo gaussiano e independiente en las coordenadas afines de los puntos. En muchas situaciones prácticas puede que el algoritmo de seguimiento se equivoque y empareje puntos en las imágenes que no se corresponden con el mismo punto físico. Estas parejas son los llamados *outliers* de la distribución de ruido gaussiano. Estas parejas de puntos pueden ser perjudiciales en la estimación de la homografía, lo que justifica que deban ser identificados.

El objetivo del presente algoritmo es determinar los *inliers* a partir de las parejas dadas y estimar la homografía consistente con los *inliers* gracias a los algoritmos presentados en secciones anteriores. Se habla de *estimación robusta* porque es tolerante con los *outliers* (observaciones que siguen una distribución de error distinta a la esperada).

Resumimos el algoritmo en los siguientes pasos: dadas las correspondencias de puntos entre imágenes $(\mathbf{x}_i, \mathbf{x}_i')$, $i = 1, \ldots, n$

1. **RANSAC básico**: Repetir hasta haber probado N muestras (donde N es determinado adaptativamente según el algoritmo 3.5 de [1, pág. 105]) o haber logrado más de $T = (1 - \epsilon_{\text{fijo}})n$ puntos consistentes con el modelo (*inliers*).

   a) Seleccionar una muestra aleatoria de 4 parejas de puntos y calcular la homografía $\mathtt{H}$ mediante el algoritmo lineal (solución exacta).

   b) Calcular la distancia $d_\perp$ de cada punto al modelo de homografía indicado por $\mathtt{H}$.

   c) Calcular el número de puntos consistentes con el modelo $\mathtt{H}$ como el número de parejas de puntos para los que $d_\perp < t = \sqrt{5{,}99}\sigma$ píxeles, siendo σ la desviación típica del ruido gaussiano esperado.

   d) Comparar: Retener la $\mathtt{H}$ con el mayor soporte (mayor número de *inliers*). En el caso de empate, retener la solución con la menor desviación típica de *inliers* (en cuanto a distancia al modelo $d_\perp$).

   e) Estimar la proporción ϵ de puntos no consistentes (*outliers*) y el número de muestras aleatorias a probar, N.

2. **Estimación óptima no lineal**: re-estimar $\mathtt{H}$ minimizando el error de reproyección a partir del mayor y mejor soporte encontrado. Se utiliza el Gold Standard sólo sobre las parejas clasificadas como consistentes con el modelo.

3. **Identificación de nuevas parejas** consistentes con el modelo, igual que en los pasos c) y d). Continuar minimizando la distancia geométrica (paso 2) hasta que el soporte deje de crecer.

El valor de ϵ_{fijo} indica el tanto por uno de *outliers* esperados. Es elección del autor y se utiliza para determinar con qué proporción de *inliers* se pasa a optimizar mediante el Gold Standard. Valores típicos: 0.2 a 0.4.

La distancia $d_\perp$ de cada punto al modelo es el error de transferencia simétrico de cada pareja de puntos. Aunque para ser consistente con la función de coste optimizada deberían utilizarse otras medidas, como el error de reproyección, o la aproximación de Sampson al mismo.

El valor del ruido esperado σ se fija a priori y no se modifica durante el resto de operaciones. Dado que el algoritmo puede trabajar con coordenadas de píxel o con coordenadas a las que se ha aplicado una normalización afín, lo lógico es poner un valor de σ adaptativo. En la práctica se mide la longitud del lado mayor del menor rectángulo que contiene a los puntos en las imágenes (*bounding box*) y se asigna a σ un tanto por ciento de esa longitud, típicamente: $1\,\% - 3\,\%$. En caso de utilizar coordenadas no normalizadas y que este valor de σ supere los 10 píxeles (una imagen muy grande), se acota: $\sigma = 10$.

Umbral de clasificación

El umbral t que indica si un punto es o no consistente con el modelo se suele determinar empíricamente, aunque es posible analizarlo desde el punto de vista estadístico. Se desea elegir el umbral de distancia, t, tal que con una probabilidad α el punto bajo estudio es consistente con el modelo. Normalmente se elige $\alpha = 0{,}95$, es decir, un *inlier* será incorrectamente rechazado un 5 % de las veces.

El cálculo del umbral requiere el conocimiento de la función densidad de probabilidad de la variable aleatoria (v.a.) que es la distancia del modelo a un punto consistente con él. Esta distribución es conocida bajo la hipótesis de ruido aditivo que venimos considerando: una gaussiana de media cero y desviación típica σ. El cuadrado de la distancia al modelo $d_\perp^2$ es una suma de cuadrados de v.a. gaussianas. Si las normalizamos ($\sigma = 1$), $d_\perp^2$ sigue una distribución χ_ν^2 con ν grados de libertad. Nos interesa conocer el valor de la v.a. χ_ν^2 para el que la función de distribución acumulada vale α, $F_\nu(k^2) = \alpha \Leftrightarrow P(k^2 \leq \alpha)$. El valor de $k^2 = t^2/\sigma^2$ indica el valor del umbral de clasificación en la distancia.

$$\begin{cases} \text{punto consistente} & (\textit{inlier}) & d_\perp^2 & < & t^2 \\ \text{punto inconsistente} & (\textit{outlier}) & d_\perp^2 & \geq & t^2 \end{cases} \quad \text{con} \quad t^2 = k^2\sigma^2 = F_\nu^{-1}(\alpha)\sigma^2$$

Veamos un ejemplo:

Sea $\alpha = 0{,}95$ y tomemos varios valores para los grados de libertad: $\nu = 1, 2, 3, 4$. Los valores de la función de distribución de una v.a. χ_ν^2 están tabulados (si no, se obtienen con el comando `chi2inv` de MATLAB) y resulta:

ν	1	2	3	4
k^2	3.8415	5.9915	7.8147	9.4877

Según [1], para una recta, $\nu = 1$, sólo se mide la distancia perpendicular a la misma. Para un punto, $\nu = 2$, en la distancia a otro punto intervienen ambas coordenadas x e y. En el algoritmo original de [1] se establece $k^2 = 5{,}9915$, es decir, $\nu = 2$. En mi opinión, considerando que cada correspondencia de puntos son 2 puntos por imagen, cada uno con 2 coordenadas, el número de variables aleatorias gaussianas al cuadrado que se suman para formar $d_\perp^2$ son $\nu = 4$, y por lo tanto, $k^2 = 9{,}4877$. Para no contradecir a Hartley, en la implementación establecemos su umbral.

4.5. Aplicación: calibración de una cámara rotatoria

Las homografías del plano sirven para muchas cosas: corregir la distorsión perspectiva de una imagen (rectificación), crear imágenes panorámicas a partir de varias imágenes, etc. A continuación presentamos una aplicación destinada a obtener información sobre la cámara que adquirió las imágenes, entre

las cuales establecemos homografías.

Supongamos que mediante una cámara de píxeles cuadrados se obtienen varias imágenes de una escena estática sin variar el centro óptico de la cámara. Es posible recuperar la matriz de parámetros intrínsecos de la cámara rotatoria. El algoritmo para hacerlo se resume en cuatro pasos:

Dadas unas correspondencia de puntos, $\mathbf{x}_i, \mathbf{x}'_i, \mathbf{x}''_i \ldots$, entre $m \geq 3$ imágenes,

1. Estimar las homografías entre pares de imágenes.

2. Calcular los puntos correspondientes a $(i, \pm 1, 0)^\top$ en cada una de las imágenes, es decir, transferir todos los puntos que son proyección de Ω_∞ a cada imagen (En total se obtienen $2m$ puntos en cada una).

3. Ajustar una cónica a cada conjunto de puntos en cada imagen. Tomamos esta cónica como la **p**royección de la **c**ónica **a**bsoluta: $\omega = \mathrm{PAC}$ (IAC en inglés: image of the absolute conic).

4. Calcular la matriz de parámetros intrínsecos $\mathtt{K}$ de la cámara como descomposición Cholesky de la matriz de la cónica $\omega = (\mathtt{K}\mathtt{K}^\top)^{-1}$.

Se necesitan, al menos, tres imágenes para obtener seis puntos de ω en cada imagen y así ajustar una cónica (para definir unívocamente una cónica hacen falta cinco puntos). Y para definir la homografía entre cada par de imágenes hacen falta, al menos, cuatro puntos.

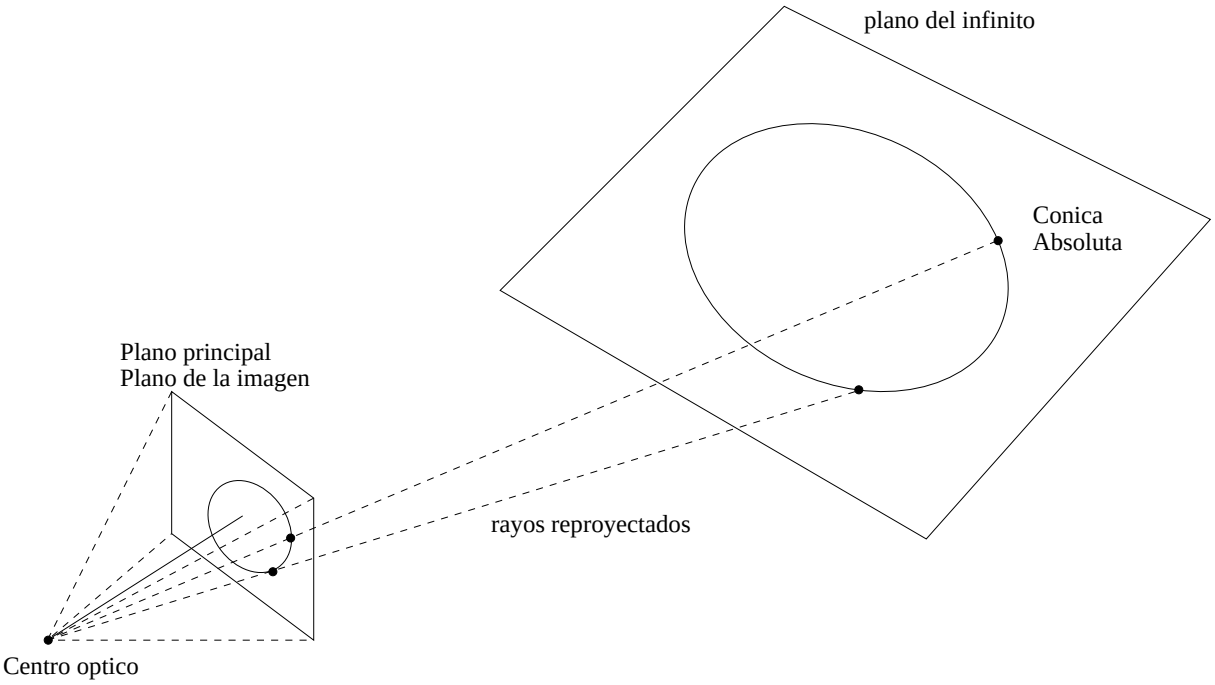


Figura 4.3: Rectas del *Calibration Pencil*. Si se conoce la forma de los píxeles, se conocen dos rectas que cortan a la cónica absoluta

Fundamentos del algoritmo

La matriz de proyección de cada imagen de una cámara rotatoria es $\mathtt{P}^i = \mathtt{K}^i[\mathtt{R}^i \mid \mathbf{0}]$. Al aplicarla a las coordenadas homogéneas de un punto del espacio $\mathbf{Q}_h = (x, y, z, t)^\top$, queda la proyección del punto $\mathbf{Q}$

en la imagen i-ésima:

$$\mathbf{q}_i \sim \mathtt{P}_i\mathbf{Q}_h = \mathtt{K}^i[\mathtt{R}^i \mid \mathbf{0}] \begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} = \mathtt{K}^i\mathtt{R}^i \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathtt{K}^i\mathtt{R}^i\mathbf{Q}_e$$

siendo $\mathbf{Q}_e$ las coordenadas euclídeas.

Supongamos que disponemos de la proyección en otra imagen: $\mathbf{q}_j \sim \mathtt{K}^j\mathtt{R}^j\mathbf{Q}_e$ y despejemos en la imagen i: $\mathbf{Q}_e \sim (\mathtt{R}^i)^{-1}(\mathtt{K}^i)^{-1}\mathbf{q}_i$. La matriz de la homografía de la imagen i a la j se obtiene de la igualdad: $\mathbf{q}_j \sim \mathtt{K}^j\mathtt{R}^j(\mathtt{R}^i)^{-1}(\mathtt{K}^i)^{-1}\mathbf{q}_i = \mathtt{H}_{ij}\mathbf{q}_i$, es decir,

$$\mathtt{H}_{ij} \sim \mathtt{K}^j\mathtt{R}^j(\mathtt{R}^i)^{-1}(\mathtt{K}^i)^{-1} \sim \mathtt{K}^j\mathtt{R}^{ji}(\mathtt{K}^i)^{-1}.$$

La homografía es la conjugada de una rotación a través de las matrices de parámetros intrínsecos. Al calibrar una cámara rotatoria no se puede recuperar la profundidad del punto euclídeo. Sea cual sea el valor de t se obtiene el mismo punto proyectado. A lo más que puede aspirar esta calibración es a recuperar $(x, y, z)^\top$, salvo factor de proporcionalidad.

Supongamos que conocemos 2 grados de libertad de la matriz de parámetros intrínsecos: la relación de aspecto de los píxeles $\tau = \alpha_u/\alpha_v$ y el ángulo entre sus lados θ (skew).

$$\mathtt{K} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_u & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Obsérvese que las rectas reproyección de los puntos $(1, \pm i, 0)^\top$ cortan a la cónica absoluta en el plano del infinito. Si $\mathbf{Q}_h = (x, y, z, 0)^\top$ son las coordenadas del punto de corte de una de las rectas con la cónica absoluta, entonces se cumple $(1, \pm i, 0)^\top \sim \mathtt{P}_i\mathbf{Q}_h = \mathtt{K}^i\mathtt{R}^i(x, y, z)^\top$, así que $(x, y, z)^\top \sim \mathtt{R}^{i\top}(\mathtt{K}^i)^{-1}(1, \pm i, 0)^\top$ y entonces, $x^2 + y^2 + z^2 = (x, y, z)(x, y, z)^\top = 0$. Estas dos rectas son las que se utilizan para definir el *Calibration Pencil* de [10] y están representadas en la figura 4.3.

En cada imagen se conocen 2 puntos que pertenecen a la proyección de la cónica absoluta (PAC o IAC), transfiriendo estos dos puntos a las otras imágenes a través de las homografías conseguimos tener varios puntos de la PAC en cada imagen. Con cinco o más puntos en cada imagen podemos estimar una cónica, que es la candidata a ser la PAC de esa cámara: $\omega^i = (\mathtt{K}^i\mathtt{K}^{i\top})^{-1}$.

4.6. Evaluación experimental

Una vez que se han descrito los algoritmos es el momento de probar experimentalmente su rendimiento. Hay dos alternativas claras: utilizar datos sintéticos o utilizar datos reales. Por ahora, consideraremos la primera opción, ya que permite un mejor estudio estadístico. A continuación expondremos la metodología empleada, no sólo para este algoritmo, sino para todos, en general.

Si suponemos datos sintéticos, conocemos las verdaderas posiciones de los puntos en las imágenes, $\bar{\mathbf{x}}$ a las que llamamos datos *exactos* y las posiciones corrompidas por el ruido, $\mathbf{x}$, a las que llamamos datos *ruidosos*. Adicionalmente, en algunos algoritmos, como el Gold Standard (Error de reproyección en las dos imágenes), también conocemos una estimación de las posiciones de los puntos, $\hat{\mathbf{x}}$, a las que a veces llamaremos *puntos estimados* o *puntos corregidos*.

Es difícil elegir un criterio para comparar los distintos algoritmos entre sí, ya que no todos los algoritmos estiman los puntos $\hat{\mathbf{x}}$, sino sólo la transformación que "mejor se ajusta" a los datos ruidosos que conoce. Más bien es al contrario, sólo se conocen las estimaciones $\hat{\mathbf{x}}$ para el caso del algoritmo

Gold Standard. Cada algoritmo minimiza una determinada función de coste, a la que llamaremos *coste propio*.

Para la estimación de homografías, la simulación consiste en la generación de un número variable de correspondencias de puntos *exactos* entre dos imágenes, $\bar{\mathbf{x}}_i \leftrightarrow \bar{\mathbf{x}}'_i$, relacionados mediante una homografía $\bar{\mathtt{H}}$, de tal forma que se verifica $\bar{\mathbf{x}}'_i = \bar{\mathtt{H}}\bar{\mathbf{x}}_i$ hasta la precisión que permite MATLAB. A continuación se añade ruido gaussiano de media cero y varianza conocida a las coordenadas afines x e y de los datos exactos. Los puntos resultantes son los que hemos llamado puntos *ruidosos*, $\mathbf{x}_i, \mathbf{x}'_i$. Estos puntos se pasan al algoritmo de estimación de la homografía, cuyos resultados evalúan cuán cerca el modelo estimado está de los datos *ruidosos* o cuán cerca está de los datos *exactos*. Un algoritmo es mejor cuanto más se acerca a los datos exactos si sólo conoce los datos ruidosos. Estos experimentos se realizan un número suficiente de veces para que sean estadísticamente significativos.

Es posible calcular analíticamente cotas del rendimiento de los algoritmos de máxima verosimilitud. Los costes propios de los algoritmos programados deberán ser comparados con dichos límites.

Error residual y error de estimación

En un problema general de estimación suele haber una función $f : \mathbb{R}^M \to \mathbb{R}^N$ (modelo), como se ha descrito en § 4.3. Donde $\mathbb{R}^M$ es el espacio de parámetros y $\mathbb{R}^N$ es el espacio de medidas. Para los datos exactos, $\bar{\mathbf{X}} \in \mathbb{R}^N$ existe un vector de parámetros $\bar{\mathbf{P}} \in \mathbb{R}^M$ tal que $\bar{\mathbf{X}} = f(\bar{\mathbf{P}})$.

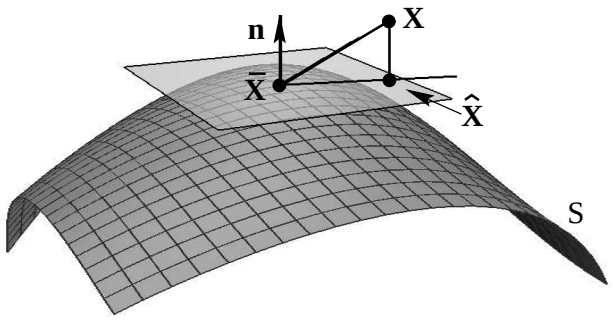


Figura 4.4: Geometría del espacio de medidas

Sea $\mathbf{X} = \bar{\mathbf{X}} + \mathcal{N}_\sigma$ un vector de medidas que ha sido obtenido sumando a los datos exactos $\bar{\mathbf{X}}$ un vector de variables aleatorias gaussianas de media cero y varianza conocida $\mathcal{N}_\sigma$. Suponemos que todas las variables aleatorias siguen la misma distribución y son independientes. Cada componente del vector $\mathcal{N}_\sigma$ tiene una varianza σ^2, así que el vector $\mathcal{N}_\sigma$ tiene una varianza $N\sigma^2$. Según varía el vector de parámetros $\mathbf{P}$ cerca del vector exacto $\bar{\mathbf{P}}$, el valor de la función $f(\mathbf{P})$ describe la superficie de medidas permitidas, de dimensión igual al número de grados de libertad (parámetros independientes) del vector de parámetros, d.

Dado un vector de medidas ruidoso $\mathbf{X}$, la estimación de máxima verosimilitud $\widehat{\mathbf{X}}$ es el punto de la superficie más cercano a $\mathbf{X}$. Bajo la hipótesis de una superficie localmente plana en un entorno de $\bar{\mathbf{X}}$, podemos aproximarla por el plano tangente y utilizar el teorema de la proyección ortogonal. En esta aproximación, la estimación de máxima verosimilitud $\widehat{\mathbf{X}}$ es la proyección de $\mathbf{X}$ sobre el plano tangente. La figura 4.4 ilustra esta aproximación.

Llamamos *error residual* a la distancia del punto estimado $\widehat{\mathbf{X}}$ al punto ruidoso $\mathbf{X}$ y llamamos *error de estimación* a la distancia del punto estimado $\widehat{\mathbf{X}}$ al punto exacto $\bar{\mathbf{X}}$.

En el problema considerado, pasamos a citar los límites teóricos (cotas de Cramer-Rao) del mínimo

error obtenible utilizando algoritmos de máxima verosimilitud.

- El mínimo error residual es

$$\epsilon_{\text{res}}^2 = E\left[\frac{\|\widehat{\mathbf{X}} - \mathbf{X}\|^2}{N}\right] = \sigma^2\left(1 - \frac{d}{N}\right) \tag{4.3}$$

- El mínimo error de estimación es

$$\epsilon_{\text{est}}^2 = E\left[\frac{\|\widehat{\mathbf{X}} - \bar{\mathbf{X}}\|^2}{N}\right] = \sigma^2\frac{d}{N} \tag{4.4}$$

donde $E[\cdot]$ es el operador esperanza matemática. En este caso, se utiliza para hallar la media de los errores para todos los experimentos. A veces se utiliza la raíz cuadrada de las ecuaciones anteriores, (lo que se conoce como valor RMS (*root-mean-square*) del error), ya que los algoritmos de máxima verosimilitud están identificados con parte de los algoritmos de funciones de coste geométrico; luego $\boldsymbol{\epsilon}_{\text{res}}$ y $\boldsymbol{\epsilon}_{\text{est}}$ se miden en píxeles.

La ventaja de utilizar como parámetro la raíz cuadrada de la media de la suma de distancias al cuadrado, $\sqrt{E[\sum_i d_i^2]}$, frente a la media de las distancias $E[\sqrt{\sum_i d_i^2}]$, es que las ecuaciones indican que el primero (error RMS) es *proporcional* a la desviación típica de ruido gaussiano σ introducido en cada coordenada de los puntos exactos, mientras que el segundo no: varía según la forma de la función raíz cuadrada. Es más cómodo ver si una función se aleja de una recta que ver si se aleja de una raíz cuadrada.

Aplicación a la estimación de homografías

El algoritmo de estimación que minimiza el error en una imagen puede considerarse como el "Gold Standard" para la situación planteada. Como se indicó en § 4.3.1, $M = 9$, pero sólo $d = 8$ parámetros son independientes y $N = 2n$, siendo n el número de correspondencias. Para este problema:

$$\begin{aligned} \epsilon_{\text{res}}^2 &= \sigma^2\left(1 - \frac{d}{N}\right) &&= \sigma^2\left(1 - \frac{4}{n}\right) \\ \epsilon_{\text{est}}^2 &= \sigma^2\frac{d}{N} &&= \sigma^2\frac{4}{n} \end{aligned} \tag{4.5}$$

Para el "verdadero" algoritmo de máxima verosimilitud, el que minimiza el error de reproyección en ambas imágenes (Gold Standard), $M = 9 + 2n$, pero sólo $d = 8 + 2n$ son independientes y $M = 4n$, como se cita en § 4.3.1. Los límites para este problema son:

$$\begin{aligned} \epsilon_{\text{res}}^2 &= \sigma^2\left(\frac{1}{2} - \frac{2}{n}\right) \\ \epsilon_{\text{est}}^2 &= \sigma^2\left(\frac{1}{2} + \frac{2}{n}\right) \end{aligned} \tag{4.6}$$

Si nos olvidamos de la variación de los errores RMS con la desviación típica del ruido, ya que es lineal, las curvas que representan el error RMS normalizado por σ ($\boldsymbol{\epsilon}_{\text{res}}/\sigma$ y $\boldsymbol{\epsilon}_{\text{est}}/\sigma$) están recogidas en la figura 4.5.

Como se puede apreciar en la figura y en las ecuaciones, para el error en una imagen, a medida que se utilizan más puntos, la distancia de los puntos estimados a los puntos exactos tiende a cero, y la distancia de los puntos estimados a los puntos ruidosos tiende a 1 (σ, si desnormalizamos el eje vertical).

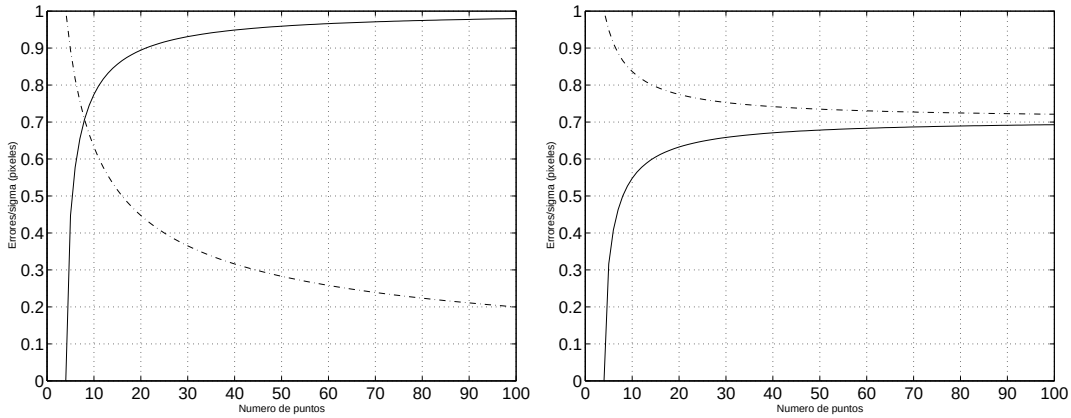


Figura 4.5: Límites teóricos de los errores RMS residual (línea continua) y de estimación (línea discontinua) para los algoritmos de estimación de la homografía según el criterio de máxima verosimilitud en 1 imagen (izquierda) y 2 (derecha)

En cambio, si se supone que hay ruido en las medidas de ambas imágenes, el valor RMS del error de estimación del algoritmo que minimiza el error de reproyección en las dos imágenes tiene un valor límite de $1/\sqrt{2}$ ($\sigma/\sqrt{2}$), no nulo.

Como se mostrará, las curvas obtenidas de forma práctica se ajustan bastante bien a las teóricas.

Descripción de los experimentos

Según la figura 4.6, los datos sintéticos utilizados para la evaluación de los algoritmos de estimación de homografías simulan dos cámaras con una distancia focal de 20 mm (ej. Sony Cyber-shot DSC-F717), píxeles cuadrados y punto principal en el origen de coordenadas, apuntando a un conjunto de puntos uniformemente distribuidos en un cuadrado de lado 2 m contenido en un plano. Las posiciones de las cámaras (centros ópticos) son aleatorias sobre una superficie esférica, situadas a una distancia media de 8 m del centro del cuadrado. Según se observa en dicha figura, las cámaras no apuntan al centro del cuadrado, sólo ligeramente.

Los puntos exactos sobre las imágenes se obtienen como resultado de proyectar los puntos 3D. Las imágenes tienen unas dimensiones medias de 1500 píxeles de lado. Los puntos ruidosos se logran sumando ruido gaussiano de media cero y desviación típica σ conocida a los exactos.

Para evitar configuraciones peligrosas, como son las homografías casi singulares debidas a que una de las cámaras vea de perfil el plano donde están los puntos, se realiza una validación de los puntos proyectados antes de utilizarlos para evaluar homografías: se ajusta una recta por mínimos cuadrados a cada proyección de puntos y se mide la distancia de los puntos a la recta. El criterio de validación se basa en la proporción de puntos cuya distancia a la recta es menor que un umbral. Por ejemplo, si el número de puntos, cuya distancia a la recta es inferior a 30 píxeles, es menor que el 40 % del total de puntos, entonces esa proyección no es degenerada. Se hace lo mismo con las dos proyecciones y se valida la configuración si ninguna de las proyecciones es degenerada.

Los algoritmos que se van a comparar son: el algoritmo lineal con normalización afín de los datos (DLT−NA), junto con su modificación por el autor (DLT−NAG) y los tres algoritmos geométricos: error en una imagen (GS1 - Gold Standard 1 imagen), error de transferencia simétrico (TS) y error de reproyección (GS2 - Gold Standard 2 imágenes).

Para cada algoritmo se evalúa cómo evoluciona su coste propio. Esto es, cómo varía la mínima distancia

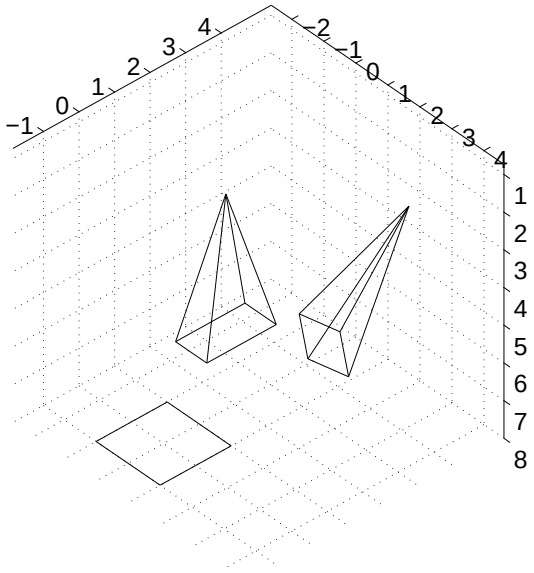


Figura 4.6: Disposición de las cámaras y los puntos 3D generados para los experimentos de estimación de homografías

algebraica *normalizada* en los algoritmos lineales y los costes geométricos en los otros algoritmos. Por ejemplo, para el error en una imagen, se mide la media de los costes que devuelve el algoritmo, para cada valor de ruido y número de puntos.

$$E\left[\frac{1}{2n}\sum_i d(\mathbf{x}'_i, \mathtt{H}\bar{\mathbf{x}}_i)^2\right].$$

Este valor hay que compararlo con el valor de $\boldsymbol{\epsilon}^2_{\text{res}}$ de (4.5). También se mide

$$E\left[\frac{1}{2n}\sum_i d(\bar{\mathbf{x}}'_i, \mathtt{H}\bar{\mathbf{x}}_i)^2\right],$$

un valor que no lo devuelve la función de coste del algoritmo, pero se puede calcular si se conoce la homografía. Este valor se compara con $\boldsymbol{\epsilon}^2_{\text{est}}$ de (4.5). Para el error de reproyección, se calculan

$$E\left[\frac{1}{4n}\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2\right], \qquad E\left[\frac{1}{4n}\sum_i d(\bar{\mathbf{x}}_i, \hat{\mathbf{x}}_i)^2 + d(\bar{\mathbf{x}}'_i, \hat{\mathbf{x}}'_i)^2\right].$$

El primero es el coste que devuelve el algoritmo, que se debe comparar con $\boldsymbol{\epsilon}^2_{\text{res}}$ en (4.6) y el segundo se puede calcular a partir de los puntos corregidos y la homografía; debe ser comparado con $\boldsymbol{\epsilon}^2_{\text{est}}$ en (4.6).

En los algoritmos algebraicos es posible comparar las menores distancias algebraicas (menores valores singulares). Esta comparación no es muy fiable porque no es geométrica, pero como se verá en las pruebas, la mínima distancia algebraica aumenta linealmente con la desviación típica de ruido. Esa distancia informa sobre "cuánto se cumplen las ecuaciones propias del sistema". Estas ecuaciones dependen de las coordenadas de los puntos, por lo que parece lógico que su cumplimiento varíe de forma lineal con el ruido. Además, es mejor comparar distancias algebraicas normalizadas, σ^2_{min}/n, que distancias algebraicas absolutas. También parece lógico, ya que el vector de error $\epsilon = \mathtt{A}\mathbf{h}$ aumenta de tamaño con el número de puntos, así que el coste $\sigma^2_{\text{min}} = \|\epsilon\|^2 = \sum_{i=1}^n \|\epsilon_i\|^2$ también aumenta con

el número de puntos.

No es fácil elegir un parámetro de calidad para comparar todos los algoritmos entre sí, ya que la mayoría sólo estima la homografía, sin incluir las posiciones de las verdaderas correspondencias (puntos corregidos) como hace el GS2. Si poseemos como único elemento estimado la homografía $\mathtt{H}$, el criterio de comparación que se nos ocurre es el error de transferencia simétrico, tanto respecto de los datos ruidosos como respecto de los datos exactos. Sin embargo, este criterio no sigue las curvas teóricas previamente descritas del comportamiento de los algoritmos de máxima verosimilitud. Las ecuaciones de estos parámetros son:

$$\begin{aligned} PQ^2_{\text{res}} &= \frac{1}{4n}\sum_{i=1}^{n}\left[d(\mathbf{x}_i, \mathtt{H}^{-1}\mathbf{x}'_i)^2 + d(\mathbf{x}'_i, \mathtt{H}\mathbf{x}_i)^2\right] \\ PQ^2_{\text{est}} &= \frac{1}{4n}\sum_{i=1}^{n}\left[d(\bar{\mathbf{x}}_i, \mathtt{H}^{-1}\mathbf{x}'_i)^2 + d(\bar{\mathbf{x}}'_i, \mathtt{H}\mathbf{x}_i)^2\right]. \end{aligned} \tag{4.7}$$

El término $1/4n$ normaliza el coste para cada coordenada del vector de medidas.

Los valores experimentales elegidos para la desviación típica de ruido σ son 11: de 0 a 5 en pasos de 0.5. El número de experimentos realizados para que las medias se estabilicen es $n_{\text{exp}} = 200$. Además, las pruebas incluyen una variación del número de puntos, dentro de los permitidos ($n \geq 4$), los cuales son $n = \{5, 10, 15, 20, 40, 70, 100\}$. No se evalúan los algoritmos con $n = 4$ puntos ya que la solución es exacta para dicho caso.

A continuación presentaremos varias gráficas de resultados experimentales. El convenio seguido en la representación es utilizar línea continua para los errores residuales (distancia de los puntos estimados a los *ruidosos*) y línea discontinua para los errores de estimación (distancia de los puntos estimados a los *exactos*). Algunas gráficas incluyen, además del valor medio de la variable aleatoria bajo estudio, el valor de la mediana. En tales gráficas, el valor medio se representa por un pequeño círculo (○), y la mediana por una cruz (+).

Costes propios

Comencemos por los costes *propios*, en concreto por ver cómo se cumplen las ecuaciones teóricas para los algoritmos de máxima verosimilitud en 1 (GS1) y 2 imágenes (GS2). Dejamos para el final el comentario sobre la variación de la mínima distancia algebraica normalizada en función del ruido.

Como es lógico, en esta memoria no se incluyen todas las gráficas. En este caso, se han seleccionado las gráficas para $n = \{5, 20, 100\}$ puntos, ya que parecen valores interesantes donde comparar los resultados con los esperados según las curvas teóricas de la figura 4.5.

Una aclaración más: en la evaluación práctica del algoritmo GS1 se han utilizado como datos en la primera imagen los datos ruidosos para comparar con otros algoritmos y los datos exactos para comparar con las curvas teóricas.

La figura 4.7 muestra los resultados experimentales de los errores RMS residual $\boldsymbol{\epsilon}_{\text{res}}$ y de estimación $\boldsymbol{\epsilon}_{\text{est}}$ de los algoritmos de máxima verosimilitud en 1 y 2 imágenes, para $n = 5$ puntos. Las líneas con círculos son los valores medios y las líneas con cruces son las medianas, como ya se ha indicado. La figura 4.8 muestra los mismos resultados para $n = 20$ puntos, mientras que la figura 4.9 lo hace para $n = 100$.

Comparándolas entre sí se aprecia cómo evolucionan los costes con el número de puntos y con σ, siendo esta última aproximadamente lineal, según predice la teoría. Para el algoritmo GS1, a medida que aumenta el número de puntos se invierten las curvas: la distancia a los datos exactos disminuye y la distancia a los datos ruidosos aumenta. Además, siguen fielmente los valores indicados por las

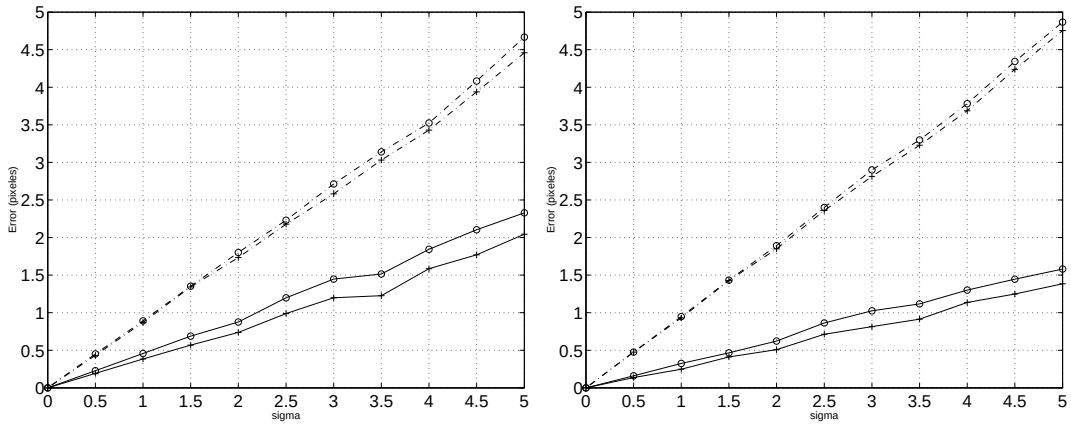


Figura 4.7: Resultados experimentales: errores RMS residual y de estimación de los algoritmos GS1 (izquierda) y GS2 (derecha), $n = 5$ puntos.

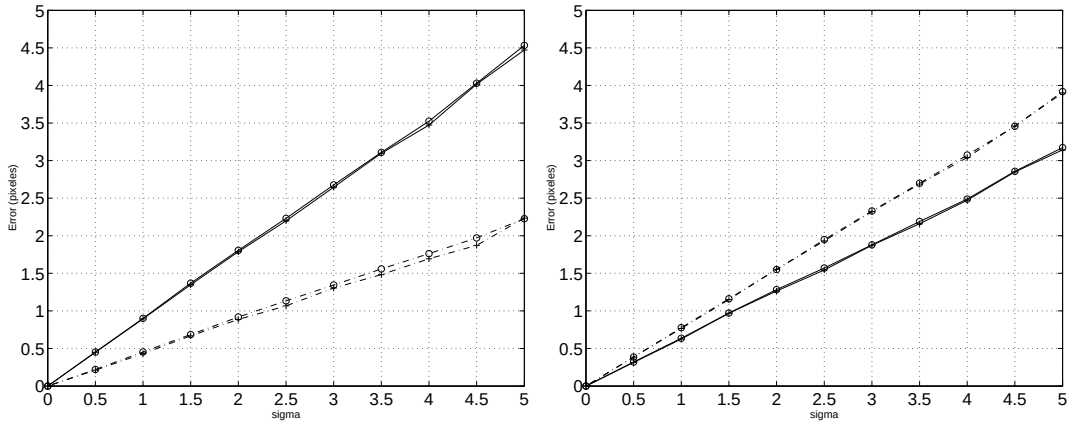


Figura 4.8: Resultados experimentales: errores RMS residual y de estimación de los algoritmos GS1 (izquierda) y GS2 (derecha), $n = 20$ puntos.

curvas teóricas. Por ejemplo, para $n = 100$ puntos, el valor experimental de $\boldsymbol{\epsilon}_{\text{res}}/\sigma$ para el GS1 es 1/5 y el valor obtenido para $\boldsymbol{\epsilon}_{\text{res}}/\sigma$ es un poco inferir a 1 (figura 4.9). Ambos valores coinciden con los valores esperados de la gráfica de la izquierda de la figura 4.5. En la misma situación, para el GS2, ambos valores están un poco por debajo y un poco por encima de 3,5/5, respectivamente. Los valores teóricos son idénticos, como muestra a gráfica de la derecha de la figura 4.5.

Pasemos a comentar brevemente la variación de la distancia algebraica normalizada (coste propio algebraico) frente al ruido, la cual está recogida en la figura 4.10.

Como se puede apreciar, la variación de $\sigma_{\min}/\sigma$ (lamentamos que se utilice la misma notación para los valores singulares y para la desviación típica) es *lineal*. Las rectas obtenidas para el resto valores de n mantienen prácticamente en la misma pendiente.

Comparación respecto del parámetro de calidad

Pasemos a comparar todos los algoritmos entre sí, mediante los parámetros de calidad escogidos: PQ_{res} y PQ_{est}, que miden los costes de transferencia simétricos respecto de datos ruidosos y datos exactos,

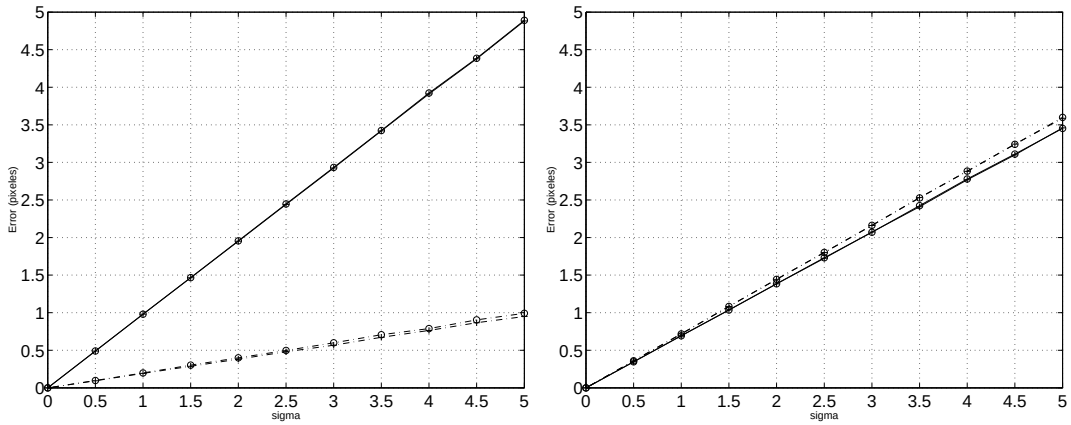


Figura 4.9: Resultados experimentales: errores RMS residual y de estimación de los algoritmos GS1 (izquierda) y GS2 (derecha), $n = 100$ puntos.

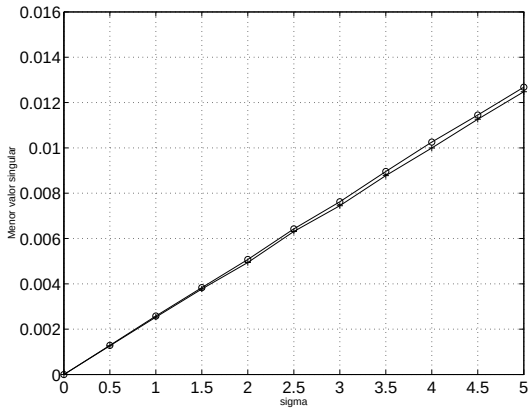


Figura 4.10: Resultados experimentales: variación del coste algebraico normalizado del algoritmo (DLT_NA) en función de la desviación típica de ruido ($n = 100$ puntos).

respectivamente.

Las gráficas 4.11 y 4.12 muestran la variación del error de transferencia simétrico para los algoritmos DLT−NA, DLT−NAG, GS1, TS y GS2, en función de la desviación típica de ruido σ y para varios valores del número de puntos n.

A grandes rasgos, podemos decir que siguen la tendencia de inversión predicha por las curvas teóricas del GS1: a medida que aumenta el número de puntos la distancia de los puntos estimados a los exactos disminuye, mientras que la distancia de los puntos estimados a los ruidosos aumenta. Sin embargo, se debe apreciar que la escala vertical no es la misma: los errores son mayores.

En general, los algoritmos DLT−NA, GS1, TS y GS2 dan resultados bastante parecidos. A pesar de que el parámetro de calidad elegido es el error de transferencia simétrico, se cumple que los mejores algoritmos son el TS y el GS2, seguidos del GS1 del DLT−NA. Además, se distingue una tendencia: conforme crece el número de puntos las estimaciones de estos algoritmos difieren menos.

También se observa la comparativa entre el algoritmo lineal modificado (DLT−NAG) y el resto: para $n = 5$ puntos las estimaciones no son buenas, pero según aumenta el número de puntos los resultados

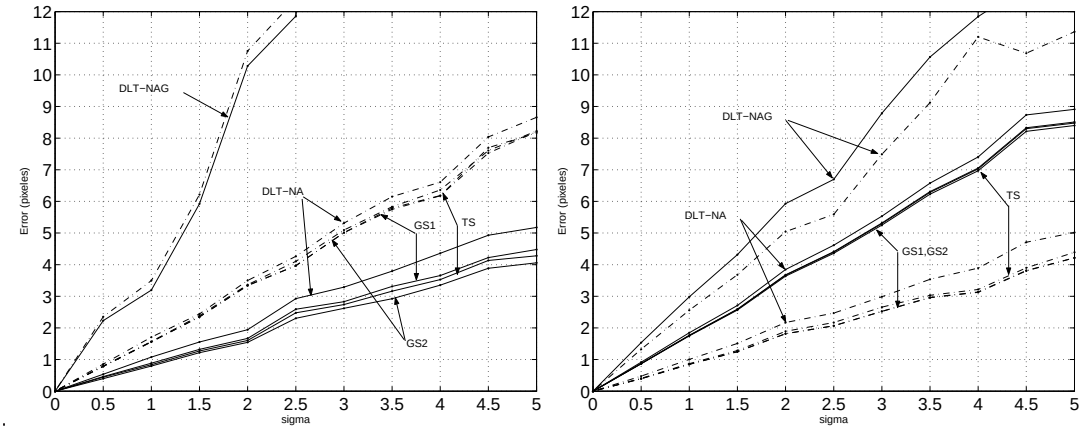


Figura 4.11: Resultados experimentales: errores de transferencia simétrico residual y de estimación normalizados en función de la desviación típica de ruido. En la imagen izquierda $n = 5$ y en la derecha, $n = 20$.

son cada vez mejores. Esta es la aplicación para la que se diseñó: muchos puntos para ejecutar los cálculos con mucho menor coste computacional que el detrimento en los resultados: es 13.5 veces más rápido y sólo el doble de malo que el resto.

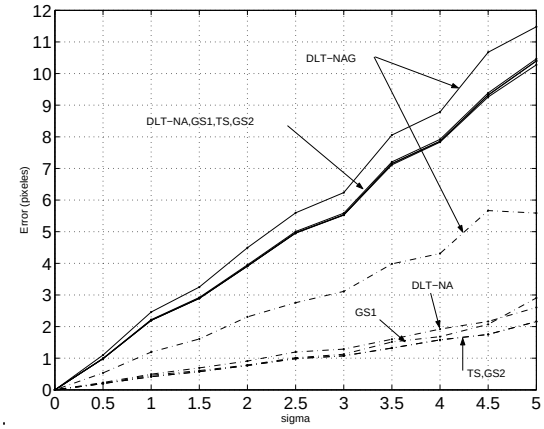


Figura 4.12: Resultados experimentales: errores de transferencia simétrico residual y de estimación normalizados en función de la desviación típica de ruido, $n = 100$ puntos.

Capítulo 5

Matriz de Proyección

5.1. Introducción

Antes de pasar a la calibración proyectiva de cámaras es necesario saber caracterizar las matrices de proyección que las representan. En este capítulo se presentan los elementos que componen una cámara proyectiva en una reconstrucción euclídea y la información recogida en la matriz de proyección de una cámara. También se incluyen varios algoritmos de estimación de dicha matriz, según lo dicho al principio del capítulo 4.

Las siguientes dos secciones hacen referencia al capítulo 5 de [1], mientras que la cuarta sección se refiere al capítulo 6. Existe un gran paralelismo entre los algoritmos de este capítulo y los del capítulo de cálculo de homografías.

5.2. Elementos de una cámara en una reconstrucción euclídea

Entender esta sección es importante porque se utiliza para optimizar una reconstrucción euclídea (ajuste de haces euclídeo § 8.2), último módulo del esquema de procesamiento visto en § 2.4.

Como se ha indicado en § 2.2.3, una matriz de proyección en un sistema de referencia euclídeo se puede descomponer de la forma:

$$\mathtt{P} = \mathtt{KR}[\mathtt{I} \mid -\widetilde{\mathbf{C}}]$$

siendo $\mathtt{K}$ la matriz de parámetros intrínsecos, $\mathtt{R}$ una matriz de rotación, $\mathtt{I}$ la matriz identidad y $\widetilde{\mathbf{C}}$ la posición del centro óptico de la cámara (traslación) en el sistema de referencia ortonormal considerado. Contemos los grados de libertad: 5 para $\mathtt{K}$, 3 para la rotación y 3 para la traslación, 11 en total. Este número coincide con los grados de libertad de una matriz de 3×4 arbitraria definida salvo proporcionalidad.

La submatriz $\mathtt{M} = \mathtt{KR}$, formada por las tres primeras columnas de $\mathtt{P}$, es regular. Esta propiedad caracteriza las matrices de proyección en un sistema de referencia euclídeo, las cuales reciben el nombre de *cámaras proyectivas finitas* (*finite projective cameras*), aunque también las llamaremos "matrices de proyección métricas". Dada la matriz $\mathtt{P}$ de una cámara proyectiva finita, es lícito descomponerla para obtener la matriz de parámetros intrínsecos $\mathtt{K}$, la orientación $\mathtt{R}$ y la posición, señalada por el centro óptico de la cámara.

$$\mathtt{P} = \mathtt{KR}[\mathtt{I} \mid -\widetilde{\mathbf{C}}] = \mathtt{M}[\mathtt{I} \mid \mathtt{M}^{-1}\mathbf{p}_4]$$

siendo $\mathbf{p}_4$ la última columna de $\mathtt{P}$. Basta con descomponer $\mathtt{M}$ en el producto de una matriz triangular superior por una matriz de rotación, operación sencilla mediante la descomposición RQ o QR de álgebra lineal.

En realidad no basta con realizar la descomposición QR para obtener las matrices $\mathtt{K}$ y $\mathtt{R}$. Hay que normalizar la matriz triangular superior de la descomposición RQ o QR para que represente una afinidad $\mathtt{K}(3,3) = 1$ y su diagonal principal sea positiva ($\alpha_u, \alpha_v > 0$). De esta manera, $\mathtt{K}$ es una matriz de parámetros intrínsecos (más restricciones que triangular superior). Las modificaciones afectan a la matriz ortogonal que acompaña a la triangular superior en la descomposición para que se preserve el producto: $\mathtt{M} \sim \mathtt{KR}$.

5.2.1. Matrices de rotación

Además de que las matrices de rotación son interesantes por sí mismas, es importante caracterizar estas matrices para encontrar parametrizaciones adecuadas de las mismas. Recordemos la definición de matriz de rotación y sus propiedades desde un punto de vista de autovalores y autovectores, para después discutir las posibles parametrizaciones. Se puede ampliar con la lectura del apéndice A.5.1 de [1, pág. 583].

Definición

Una matriz $\mathtt{R}$ (3×3) representa una rotación de elementos en $\mathbb{R}^3$ cuando verifica ser una matriz ortogonal $\mathtt{R}^\top\mathtt{R} = \mathtt{I}$ y su determinante vale la unidad $\det(\mathtt{R}) = 1$. Tanto las columnas como las filas de $\mathtt{R}$ forman bases ortogonales de $\mathbb{R}^3$. Se suele escribir $\mathtt{R} \in SO(3)$, el grupo especial ortogonal de dimensión 3.

Propiedades

Una matriz $\mathtt{R} \in SO(3)$ admite una diagonalización utilizando los autovalores y autovectores:

$$\mathtt{R} = \mathtt{V}\mathbf{\Lambda}\mathtt{V}^{-1}. \tag{5.1}$$

Los tres autovalores de la matriz $\mathbf{\Lambda}$ son $\{1, e^{i\theta}, e^{-i\theta}\}$: todos de módulo unidad, uno es real y los otros dos son complejos conjugados, siendo la fase, θ, el ángulo de giro alrededor del eje. Los correspondientes autovectores (matriz de autovectores $\mathtt{V}$) son $\{\mathbf{a}, \mathbf{I}, \mathbf{J}\}$, donde $\mathbf{a}$ es el eje de rotación, es decir, $\mathtt{R}\mathbf{a} = \mathbf{a}$ y los vectores $\mathbf{I}$ y $\mathbf{J}$ (complejos conjugados) son los puntos circulares del plano ortogonal al eje $\mathbf{a}$.

Parametrizaciones

Una matriz de rotación (de nueve elementos) se pueden parametrizar con tres números y existen distintas alternativas:

1. **Ángulos de Euler** Una rotación cualquiera se puede expresar como tres rotaciones seguidas alrededor de los ejes coordenados, aplicadas en un orden establecido:

   $$\mathtt{R} = \mathtt{R}_z^\top \mathtt{R}_y^\top \mathtt{R}_x^\top.$$

   Siendo $(\theta_x, \theta_y, \theta_z)$, los tres ángulos alrededor de los ejes coordenados, llamados ángulos de Euler, las rotaciones simples son:

   $$\mathtt{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\operatorname{sen}\theta_x \\ 0 & \operatorname{sen}\theta_x & \cos\theta_x \end{bmatrix} \quad \mathtt{R}_y = \begin{bmatrix} \cos\theta_y & 0 & \operatorname{sen}\theta_y \\ 0 & 1 & 0 \\ -\operatorname{sen}\theta_y & 0 & \cos\theta_y \end{bmatrix} \quad \mathtt{R}_z = \begin{bmatrix} \cos\theta_z & -\operatorname{sen}\theta_z & 0 \\ \operatorname{sen}\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. **Eje de rotación y ángulo de giro alrededor del eje**. La fórmula más directa de calcular la matriz de rotación asociada a un ángulo $\theta \in \mathbb{R}$ alrededor de un eje especificado por su vector

unitario $\hat{\omega} = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ es aplicar la fórmula de Rodrigues:

$$\begin{aligned} \mathtt{R} &= \exp([\hat{\omega}]_\times \theta) \\ &= \mathtt{I} + [\hat{\omega}]_\times \operatorname{sen}\theta + [\hat{\omega}]_\times^2 (1 - \cos\theta) \\ &= \begin{bmatrix} \cos\theta + \omega_x^2(1-\cos\theta) & \omega_x\omega_y(1-\cos\theta) - \omega_z \operatorname{sen}\theta & \omega_y \operatorname{sen}\theta + \omega_x\omega_z(1-\cos\theta) \\ \omega_z \operatorname{sen}\theta + \omega_x\omega_y(1-\cos\theta) & \cos\theta + \omega_y^2(1-\cos\theta) & -\omega_x \operatorname{sen}\theta + \omega_y\omega_z(1-\cos\theta) \\ -\omega_y \operatorname{sen}\theta + \omega_x\omega_z(1-\cos\theta) & \omega_x \operatorname{sen}\theta + \omega_y\omega_z(1-\cos\theta) & \cos\theta + \omega_z^2(1-\cos\theta) \end{bmatrix} \end{aligned}$$

donde

$$[\hat{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

es la matriz antisimétrica especificada por el vector unitario del eje de rotación.

Es apreciable que $[\hat{\omega}]_\times \omega = \mathbf{0}$, que significa que al aplicar la matriz de rotación, dada por la fórmula de Rodrigues, a cualquier punto del eje de rotación se obtiene el mismo punto.

$$\mathtt{R}\omega = \mathtt{I}\omega + [\hat{\omega}]_\times \omega \operatorname{sen}\theta + [\hat{\omega}]_\times^2 \omega(1 - \cos\theta) = \omega$$

El eje de rotación y el ángulo de giro de una matriz de rotación dada se obtienen de dos propiedades: 1) la traza de la matriz de rotación sólo depende del ángulo de giro y 2) el vector $(\mathtt{R} - \mathtt{R}^\top)\mathbf{u}$ es perpendicular al eje de rotación, $\forall \mathbf{u}$.

$$\operatorname{traza}(\mathtt{R}) = 1 + 2\cos\theta \tag{5.2}$$

$$[\hat{\omega}]_\times = \mathtt{R} - \mathtt{R}^\top \tag{5.3}$$

Existe una ambigüedad, ya que se obtiene la misma $\mathtt{R}$ con $\theta' = -\theta$ y $\hat{\omega}' = -\hat{\omega}$

3. **Exponencial de una matriz antisimétrica** . Siguiendo con el mismo principio que la fórmula de Rodrigues, el vector $\mathbf{v} = \hat{\omega}\theta$ de 3 componentes parametriza la matriz.

$$\mathtt{R} = \exp([\mathbf{v}]_\times) = \exp([\hat{\omega}]_\times \theta)$$

5.3. Información muy básica de una cámara

La matriz de proyección más general no es la de la cámara proyectiva finita, que tiene la restricción de regularidad de la submatriz $\mathtt{M}$. Una *cámara proyectiva general* está representada por una matriz de 3×4 arbitraria de rango 3. También tiene 11 grados de libertad. Si no tuviese rango 3, la proyección no se realizaría sobre un plano (una imagen), sino sobre una recta o un punto. Este tipo de cámaras es la que se utiliza en la calibración proyectiva.

Centro óptico

Toda matriz de 3×4 de rango 3, $\mathtt{P}$, tiene un núcleo por la derecha de dimensión 1. Sea $\mathbf{C}$ un vector tal que expande el núcleo $\mathtt{P}\mathbf{C} = \mathbf{0}$. El vector $\mathbf{C}$ es el centro óptico de la cámara, expresado en coordenadas homogéneas. El cetro óptico es el único punto del espacio para el que no existe un punto proyectado, ya que $(0,0,0,0)^\top \notin \mathbb{P}^3$.

Para una cámara proyectiva finita ($\mathtt{M}$ no es singular), ya se ha visto que el centro óptico se puede expresar $\mathbf{C} = (-\mathtt{M}^{-1}, 1)^\top$. Una cámara en el infinito ($\mathtt{M}$ es singular) es aquella cuyo centro óptico es un punto del plano del infinito canónico $\mathbf{C} = (\mathbf{d}^\top, 0)^\top$, tal que $\mathtt{M}\mathbf{d} = \mathbf{0}$.

Plano principal

El plano principal de una cámara es que que pasa por el centro óptico de la misma y es paralelo al plano de la imagen. Las proyecciones de los puntos del plano son los puntos de la recta del infinito del plano de la imagen: $\mathtt{P}\mathbf{X} = (x, y, 0)^\top \Leftrightarrow \mathbf{P}^{3\top}\mathbf{X} = 0$, siendo $\mathbf{P}^{3\top}$ la tercera fila de la matriz $\mathtt{P}$. Es decir, que el plano de coordenadas homogéneas $\mathbf{P}^{3\top}$ es el plano principal.

Planos axiales

Los planos de coordenadas homogéneas $\mathbf{P}^{1\top}$ y $\mathbf{P}^{2\top}$ (las dos primeras filas de la matriz de proyección $\mathtt{P}$) son aquellos que pasan por el centro óptico de la cámara y su intersección con el plano de la imagen son los ejes $x = 0$ e $y = 0$, respectivamente.

5.4. Estimación de la matriz de proyección

El enunciado del problema es: dado un conjunto de puntos $\mathbf{X}_i$ en el espacio y un conjunto correspondiente de puntos $\mathbf{x}_i$ en una imagen, calcular la matriz de proyección $\mathtt{P}$ tal que $\mathbf{x}_i = \mathtt{P}\mathbf{X}_i$.

En presencia de ruido no será posible cumplir la ecuación de proyección de forma exacta, por lo que se buscará la matriz de proyección $\mathtt{P}$ que mejor ajusta las proyecciones de los $\mathbf{X}_i$ respecto de los puntos $\mathbf{x}_i$.

Sólo se estudian dos algoritmos para resolver el problema: uno lineal que minimiza la distancia algebraica y otro no lineal óptimo (Gold Standard), que minimiza el error de reproyección (distancia geométrica).

Este problema es muy parecido al del cálculo de homografías, sólo cambian las dimensiones del problema. La matriz que relaciona los datos es de 3×4, en lugar de 3×3. Muchos de los detalles discutidos en § 4.2 son también aplicables al algoritmo que aquí se describe.

5.4.1. Algoritmo lineal

La ecuación de proyección homogénea $\mathbf{x}_i = \mathtt{P}\mathbf{X}_i$ significa que los vectores a ambos miembros de la igualdad son proporcionales, es decir, que su producto vectorial es nulo: $\mathbf{x}_i \times \mathtt{P}\mathbf{X}_i = \mathbf{0}$. Llamando $\mathbf{P}^{i\top}$ a la fila i-ésima de la matriz $\mathtt{P}$, podemos expresar la anterior ecuación vectorial en forma de sistema lineal en los elementos de la matriz de proyección:

$$\mathtt{A}_i\mathbf{p} = \begin{bmatrix} \mathbf{0}^\top & -w_i\mathbf{X}_i^\top & y_i\mathbf{X}_i^\top \\ w_i\mathbf{X}_i^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i^\top \\ -y_i\mathbf{X}_i^\top & x_i\mathbf{X}_i^\top & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0} \tag{5.4}$$

De las tres ecuaciones sólo dos son linealmente independientes. Los elementos de la matriz de diseño de la correspondencia de puntos $\mathtt{A}_i$, de tamaño 3×12, son cuadráticos en las coordenadas de los datos $\mathbf{x}_i, \mathbf{X}_i$. Concatenando matrices de diseño de puntos se forma la matriz de diseño del problema. Con n parejas de puntos creamos una matriz $\mathtt{A}$ de $3n \times 12$. La matriz de proyección se calcula resolviendo el sistema:

$$\mathtt{A}\mathbf{p} = \mathbf{0} \tag{5.5}$$

La matriz $\mathtt{P}$ tiene $12 - 1 = 11$ grados de libertad, por lo tanto, el número mínimo de ecuaciones linealmente independientes necesarias para resolver el sistema homogéneo (5.5) es 11. Cada punto contribuye con 2 ecuaciones linealmente independientes, lo que implica que hacen falta $5\frac{1}{2}$ correspondencias de puntos (de la última correspondencia sólo se utiliza una de las 2 ecuaciones independientes). Si esto se cumple y los puntos están en posición general, $\mathtt{A}$ será de 11×12 y de rango 11, así que su núcleo no nulo es el vector solución $\mathbf{p}$ buscado.

Solución mínimo-cuadrática

En general, se proporcionan $n \geq 6$ parejas de puntos y la matriz de diseño $\mathtt{A}$ no es de rango 11, por lo que no existe solución exacta: ningún vector $\mathbf{p}$ verifica $\mathtt{A}\mathbf{p} = \mathbf{0}$, sino que $\mathtt{A}\mathbf{p} = \epsilon$. El planteamiento inmediato es minimizar la norma del vector de error ϵ, la distancia algebraica. Como $\mathbf{p}$ está definido salvo factor de proporcionalidad, falta imponer una restricción. Por comodidad se impone $\|\mathbf{p}\| = 1$, y se utilizan los multiplicadores de Lagrange para resolver el problema de optimización:

$$\min_{\mathbf{p}} \|\mathtt{A}\mathbf{p}\| \quad \text{sujeto a} \quad \|\mathbf{p}\| = 1$$

La solución a este problema ya se vio en § 4.2. Tomando la SVD de la matriz de diseño $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^\top$, la solución $\mathbf{p}$ es el vector singular de $\mathtt{V}$ asociado al menor valor singular, σ_{12}: la última columna de $\mathtt{V}$. El coste alcanzado es el mínimo de la función de coste: $\sigma_{12} = \min \|\mathtt{A}\mathbf{p}\|$.

Para que el algoritmo sea numéricamente estable y práctico es necesario normalizar los datos, al igual que se hace en § 4.2, pero tampoco se debe descuidar la desnormalización de los resultados, para que todo encaje. El algoritmo se conoce como "The normalized Direct Linear Transformation (DLT) for the Camera Matrix P".

Como también está implementado en este proyecto el algoritmo óptimo (Gold Standard) para la estimación de la matriz de proyección, el algoritmo lineal sólo interesa para proporcionar el punto de partida de la búsqueda no lineal.

5.4.2. Algoritmo Gold Standard

El error geométrico en una imagen se puede definir como

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2$$

siendo $\mathbf{x}_i$ los puntos observados y $\hat{\mathbf{x}}_i$ los puntos proyectados $\mathtt{P}\mathbf{X}_i$. Si el ruido en las coordenadas afines es gaussiano, entonces la solución $\mathtt{P}$ de máxima verosimilitud es la que minimiza

$$\min_{\mathtt{P}} \sum_i d(\mathbf{x}_i, \mathtt{P}\mathbf{X}_i)^2$$

Minimizar esta función de coste se puede hacer mediante una función modelo en el marco común de aplicación del algoritmo LM.

Funciones modelo y de coste

La función modelo es:

$$f : \mathbf{p} \equiv \mathbf{P} \rightarrow \widehat{\mathbf{X}} \equiv (\mathtt{P}\mathbf{X}_1, \ldots, \mathtt{P}\mathbf{X}_n)$$

El vector de parámetros está formado por los 12 elementos de la matriz de proyección; es una sobreparametrización pero es la más sencilla. El vector de medidas $\widehat{\mathbf{X}}$ se forma con las coordenadas afines de los puntos proyectados y el vector de medidas objetivo $\mathbf{X}$ tiene las coordenadas afines de los puntos observados en las imágenes. La función de coste se expresa:

$$\text{coste} = \|\mathbf{X} - \widehat{\mathbf{X}}\| = \|\mathbf{X} - f(\mathbf{P})\| = \sqrt{\sum_i d(\mathbf{x}_i, \mathtt{P}\mathbf{X}_i)^2}$$

El algoritmo discutido es el 6.1 de [1, pág. 170]. Pudiera parecer que no es un algoritmo muy empleado porque habitualmente suponemos que sólo son conocidas las imágenes y se desea conocer los puntos 3D y las matrices de proyección, sin embargo es un buen algoritmo para proporcionar una calibración proyectiva inicial de muchas cámaras, como se explicará en § 6.6.3.

Derivadas exactas

Suponiendo que un punto $\hat{\mathbf{x}}_i = \mathtt{P}\mathbf{X}_i$ tiene por coordenadas homogéneas $\hat{\mathbf{x}}_{ih} = (x_i, y_i, w_i)^\top$, las coordenadas afines que componen $\widehat{\mathbf{X}}$ son:

$$\hat{\mathbf{x}}_i = \left(\frac{x_i}{w_i}, \frac{y_i}{w_i}\right)^\top$$

La matriz jacobiana 2×12 para cada punto es:

$$\mathtt{J}_i = \frac{\partial \hat{\mathbf{x}}_i}{\partial \mathbf{p}} = \frac{1}{w_i}\begin{bmatrix} \mathbf{X}_i^\top & \mathbf{0}^\top & -\frac{x_i}{w_i}\mathbf{X}_i^\top \\ \mathbf{0}^\top & \mathbf{X}_i^\top & -\frac{y_i}{w_i}\mathbf{X}_i^\top \end{bmatrix} = \begin{bmatrix} \mathtt{I} & -\hat{\mathbf{x}}_i \end{bmatrix} \otimes \frac{1}{w_i}\mathbf{X}_i^\top$$

siendo $\mathbf{X}_i^\top = (x_i, y_i, z_i, w_i)$ las coordenadas homogéneas de los puntos 3D. La matriz jacobiana (de la función modelo) para todos los puntos se obtiene concatenando en vertical matrices $\mathtt{J}_i$.

$$\mathtt{J} = \begin{bmatrix} \mathtt{J}_1 \\ \vdots \\ \mathtt{J}_n \end{bmatrix}$$

Ampliaciones

Los algoritmos vistos están enfocados al problema de estimación de una cámara proyectiva general. Una posible mejora consiste en la adaptación del algoritmo para que estime la matriz de proyección sujeta a alguna restricción de los parámetros de la cámara proyectiva finita: skew nulo ($\theta = \pi/2$), píxeles cuadrados ($\theta = \pi/2$ y $\alpha_u = \alpha_v$), punto principal conocido (más todo lo anterior), matriz $\mathtt{K}$ conocida, etc.

5.5. Evaluación experimental

Veamos cómo se aplican las nociones generales de medición de los algoritmos al caso particular de estimación de la matriz de proyección.

El algoritmo Gold Standard § 5.4.2 es el algoritmo de máxima verosimilitud y sus números son: $M = 12$, pero sólo $d = 11$ parámetros son independientes y $N = 2n$, siendo n el número de correspondencias. Según las expresiones (4.3) y (4.4):

$$\begin{aligned} \boldsymbol{\epsilon}_{\text{res}}^2 &= \sigma^2\left(1 - \frac{d}{N}\right) &&= \sigma^2\left(1 - \frac{11}{2n}\right) \\ \boldsymbol{\epsilon}_{\text{est}}^2 &= \sigma^2\frac{d}{N} &&= \sigma^2\frac{11}{2n} \end{aligned} \tag{5.6}$$

Los límites de los errores RMS de estas expresiones cuando el número de puntos tiende a infinito son σ y 0, respectivamente. Significan que la incertidumbre en la estimación es menor a medida que se utilizan más puntos, ya que suponemos que los puntos 3D son exactos. En las imágenes, la distancia de los puntos estimados $\hat{\mathbf{x}}_i = \mathtt{P}\mathbf{X}_i$ a los puntos exactos $\bar{\mathbf{x}} = \bar{\mathtt{P}}\mathbf{X}_i$ tiende a cero, y la distancia de los puntos estimados $\hat{\mathbf{x}}_i$ a los puntos ruidosos $\mathbf{x}_i$ tiende a σ.

Las curvas que representan el error RMS normalizado por σ, $\boldsymbol{\epsilon}_{\text{res}}/\sigma$ y $\boldsymbol{\epsilon}_{\text{est}}/\sigma$, frente al número de puntos son las de la figura 5.1.

Descripción de los experimentos

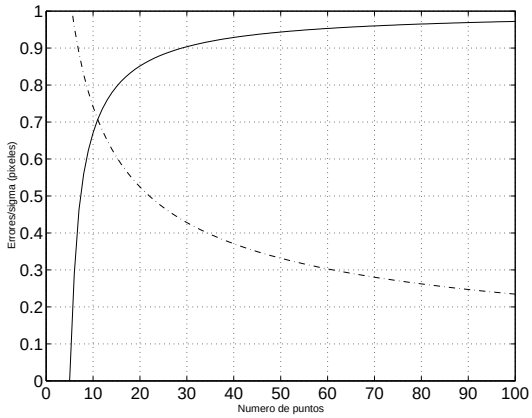


Figura 5.1: Límites teóricos de los errores RMS residual (línea continua) y de estimación (línea discontinua) del algoritmo de estimación de la matriz de proyección según el criterio de máxima verosimilitud

Los datos sintéticos utilizados para la evaluación de los algoritmos de estimación de la matriz de proyección simulan una cámara con una distancia focal de 20 mm, píxeles cuadrados y punto principal en el origen de coordenadas, apuntando a un conjunto de puntos uniformemente distribuidos dentro de en una esfera de radio 1 m. La posición de la cámara (centro óptico) es aleatoria sobre una superficie esférica de radio 8 m concéntrica con la esfera de los puntos. La cámara está ligeramente desviada, sin apuntar al centro de la esfera de puntos.

Los puntos exactos sobre la única imagen se obtienen como resultado de proyectar los puntos 3D. Los puntos ruidosos se obtienen sumando ruido gaussiano de media cero y desviación típica σ conocida a los exactos. Como datos 3D exactos se utilizan los generados en el sistema de referencia euclídeo. En este caso no hay configuraciones peligrosas como en las homografías cuasi-singulares, ya que la esfera se proyecta en una cónica.

Claramente hay dos algoritmos a evaluar: el lineal que minimiza la distancia algebraica y que utiliza una normalización afín de los datos (DLT−NA), frente al algoritmo que minimiza el error de reproyección (GS − Gold Standard).

Para cada algoritmo se evalúa su coste propio: cómo varía la mínima distancia algebraica *normalizada* en el algoritmo lineal y cómo varía el coste geométrico *normalizado* en el Gold Standard. Este coste lleva el apellido de normalizado porque también se realiza una normalización afín previa de los puntos proyectados.

El parámetro de calidad para comparar los dos algoritmos es fácil, ya que en ambos casos sólo se estima la matriz de proyección que lleva los puntos exactos 3D a la imagen. El criterio de comparación es el error de reproyección, tanto respecto de los datos ruidosos como respecto de los datos exactos. Este criterio sí sigue las curvas teóricas de la figura 5.1. Las ecuaciones de estos parámetros son:

$$\begin{aligned} PQ^2_{\text{res}} &= \frac{1}{2n}\sum_{i=1}^{n} d(\mathbf{x}_i, \mathtt{P}\mathbf{X}_i)^2 \\ PQ^2_{\text{est}} &= \frac{1}{2n}\sum_{i=1}^{n} d(\bar{\mathbf{x}}_i, \mathtt{P}\mathbf{X}_i)^2. \end{aligned} \tag{5.7}$$

El término $1/2n$ normaliza el coste para cada coordenada del vector de medidas, porque si hay n puntos en la única imagen, hay $2n$ coordenadas.

Los valores experimentales elegidos para la desviación típica de ruido σ son los mismos que para evaluar homografías: de 0 a 5 en pasos de 0.5. También mantenemos el número de experimentos realizados $n_{\mathrm{exp}} = 200$. Las pruebas incluyen una variación del número de puntos, dentro de los permitidos ($n > 5\frac{1}{2}$), los cuales son $n = \{6, 8, 10, 15, 20, 40, 70, 100\}$. En este caso nunca hay solución exacta si se coge un número entero de correspondencias de puntos.

Costes propios

Poco incidiremos sobre los costes propios, sólo un par de comentarios: (1) las gráficas del coste propio algebraico normalizado del algoritmo lineal frente a σ, sigue una variación lineal, al igual que el equivalente algoritmo lineal de las homografías; (2) el coste propio del algoritmo geométrico es el mismo que el del parámetro de calidad, luego se verá en el siguiente párrafo.

Comparación respecto del parámetro de calidad

Las figuras 5.2 y 5.3 muestran los resultados experimentales de la variación del error de reproyección RMS frente al ruido, tanto residual $\boldsymbol{\epsilon}_{\mathrm{res}}$ como de estimación $\boldsymbol{\epsilon}_{\mathrm{est}}$ y para los valores del número de puntos seleccionado: $n = \{6, 10, 20, 100\}$. Se han elegido estos valores porque muestran la evolución de las curvas teóricas (figura 5.1) con el número de puntos. Son valores interesantes de comparación. En la representación, seguimos el mismo convenio que el de las gráficas del capítulo de homografías.

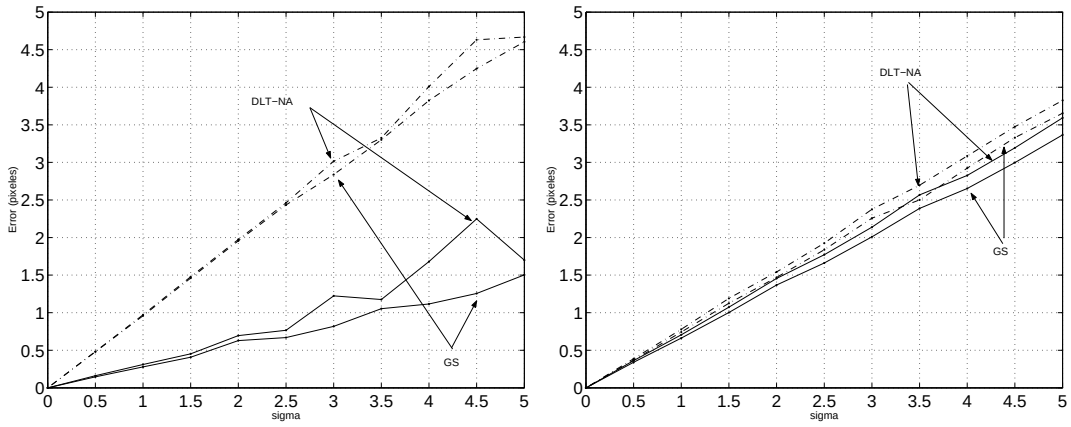


Figura 5.2: Resultados experimentales: errores RMS residual y de estimación de los algoritmos lineal y Gold Standard con $n = 6$ puntos (izquierda) y $n = 10$ (derecha).

Las diferencias de estimaciones entre los algoritmos lineal y Gold Standard son menores cuantos más puntos se utilicen y en general son bastante parecidas, siendo mejores las del Gold Standard. Analizando un poco más en detalle, para $n = 6$ (el mínimo número de correspondencias enteras necesarias) se aprecia que los resultados del algoritmo lineal son buenos, aunque de vez en cuando se generen estimaciones que dan lugar a errores un poco más grandes que provoquen una variación sustancial de la media.

También se observa el fenómeno de inversión de los errores residual y de estimación: conforme crece el número de puntos, el error residual aumenta y el error de estimación disminuye. Para $n = 10$ las curvas son casi coincidentes, como predice la curva teórica (figura 5.1).

Además, no sólo se cumple cualitativamente este fenómeno, sino también numéricamente, ajustándose fielmente los valores obtenidos a los valores esperados. Por ejemplo, para el algoritmo GS en $n = 20$ (figura 5.3), $\boldsymbol{\epsilon}_{\mathrm{est}}/\sigma$ es un poco mayor que $2{,}7/5 = 0{,}54$ y $\boldsymbol{\epsilon}_{\mathrm{res}}/\sigma \approx 4{,}25/5 = 0{,}85$, que son valores muy

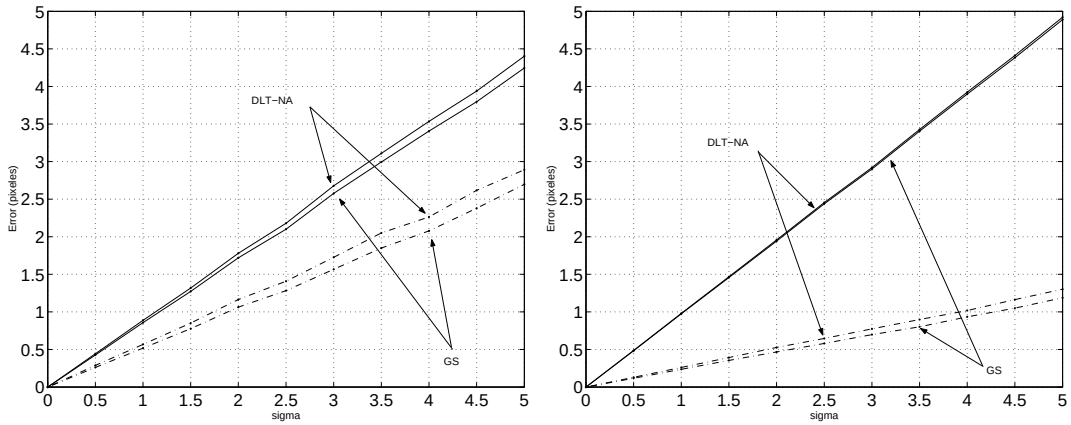


Figura 5.3: Resultados experimentales: errores RMS residual y de estimación de los algoritmos lineal y Gold Standard con $n = 20$ puntos (izquierda) y $n = 100$ (derecha)

próximos a los teóricos. El mismo juego de números se puede hacer con las otras gráficas y se justifica que las curvas experimentales se ajustan a las teóricas.

Capítulo 6

Calibración proyectiva

6.1. Introducción

El primer bloque del esquema de procesamiento introducido en § 2.4 que forma parte del dominio del proyecto es el que realiza la calibración proyectiva. En este capítulo afrontamos el problema de obtener dicha calibración a partir de las correspondencias de puntos (y a veces, rectas) entre las imágenes.

El capítulo está organizado de la siguiente manera: primero se supone la existencia de sólo 2 cámaras y se estudia la geometría de la escena, la geometría epipolar, que nace de la restricción epipolar y da lugar a relaciones bilineales entre las correspondencias de puntos en las imágenes: las matrices fundamental y esencial. A continuación se estudia la geometría de tres cámaras, que da lugar a relaciones bilineales entre parejas de cámaras y relaciones trilineales, el tensor trifocal, al considerar las tres cámaras a la vez. Seguimos aumentando el número de cámaras, esta vez a cuatro y se presenta la geometría involucrada, aparecen relaciones bilineales, trilineales y cuadrilineales entre las correspondencias de puntos: surge el tensor cuadrifocal.

Si seguimos aumentando el número de cámaras la complejidad aumenta, ya que aumenta el número de restricciones en el rango de una matriz que contiene las matrices de proyección de las cámaras y las coordenadas de los puntos observados. En tales situaciones se parametriza el problema de forma directa mediante las matrices de proyección y las coordenadas de los puntos 3D de los que provienen los puntos observados: no interesa obtener ningún tensor de mayor dimensión. Aparece la técnica del ajuste de haces como preponderante en estas situaciones.

Todo este capítulo utiliza el marco matemático de la geometría proyectiva como herramienta básica de análisis.

6.2. Geometría de dos cámaras

La presente sección trata sobre las matrices esencial y fundamental entre dos imágenes. Ambas matrices describen las relaciones bilineales entre correspondencias de puntos. La referencia básica está constituida por los capítulos 8 y 10 de [1].

Primero derivamos las ecuaciones de estas matrices en función de las correspondencias de puntos y las matrices de parámetros intrínsecos. A continuación veremos algoritmos de estimación de la matriz fundamental, ya que la mayoría de las veces desconocemos las matrices de parámetros intrínsecos de las cámaras, por lo que trabajamos en coordenadas de píxel, no en coordenadas normalizadas.

Una aplicación directa una vez conocida la matriz fundamental es la calibración proyectiva de dos cámaras, como se mostrará al final de la sección.

6.2.1. Derivación de las matrices esencial y fundamental

Supongamos que tenemos una pareja de cámaras estéreo que están viendo un punto $\mathbf{x}$ del mundo real, que se proyecta en los dos planos de la imagen en $\bar{\mathbf{x}}_1$ y $\bar{\mathbf{x}}_2$ (Como estamos tratando con coordenadas homogéneas, $\mathbf{x}$ es de 4×1, y $\bar{\mathbf{x}}_1$ y $\bar{\mathbf{x}}_2$ son de 3×1). Si asumimos que las cámaras están calibradas, entonces $\bar{\mathbf{x}}_1$ y $\bar{\mathbf{x}}_2$ están dados en coordenadas normalizadas, (cada uno está dado respecto del sistema de referencia adaptado a su cámara).

La restricción epipolar dice que el vector que va desde el centro óptico de la primera cámara hasta el punto proyectado en la primera imagen, el vector desde el segundo centro óptico al punto proyectado en la segunda imagen, y el vector desde el primer centro óptico hasta el segundo centro óptico, son todos coplanarios: forman un triángulo (figura 6.2). En coordenadas normalizadas, esta restricción se puede expresar fácilmente como:

$$\bar{\mathbf{x}}_2^\top (\mathbf{t} \times \mathtt{R}\bar{\mathbf{x}}_1) = 0,$$

donde $\mathtt{R}$ y $\mathbf{t}$ representan el movimiento rígido de rotación más traslación entre los dos sistemas de coordenadas de las cámaras. La multiplicación por $\mathtt{R}$ es necesaria para transformar $\bar{\mathbf{x}}_1$ al sistema de coordenadas de la segunda cámara.

Si definimos $[\mathbf{t}]_\times$ como la matriz tal que $[\mathbf{t}]_\times \mathbf{y} = \mathbf{t} \times \mathbf{y}$ para cualquier vector $\mathbf{y}$ [1], podemos reescribir la ecuación como una ecuación lineal:

$$\bar{\mathbf{x}}_2^\top ([\mathbf{t}]_\times \mathtt{R}\bar{\mathbf{x}}_1) = \bar{\mathbf{x}}_2^\top \mathtt{E}\bar{\mathbf{x}}_1 = 0,$$

donde $\mathtt{E} = [\mathbf{t}]_\times \mathtt{R}$ se suele denominar *matriz esencial*.

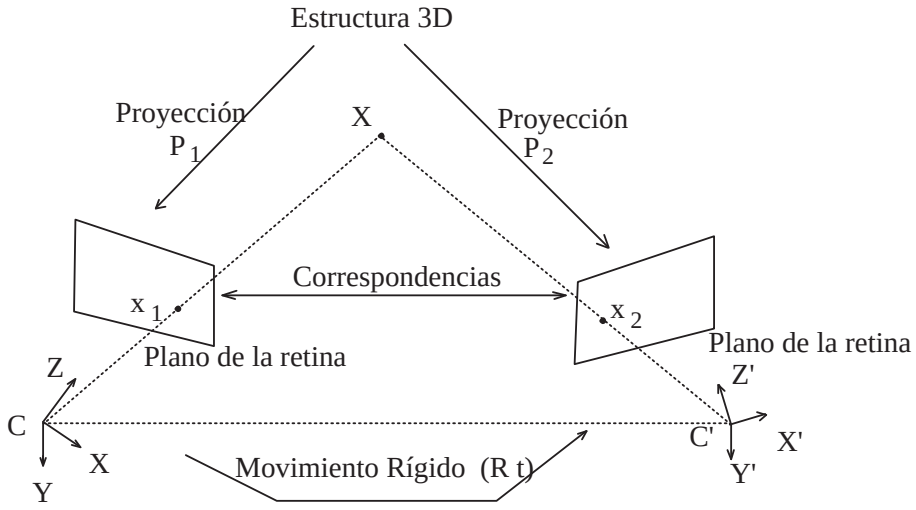


Figura 6.1: Elementos utilizados en la geometría epipolar

Ahora supongamos que las cámaras no están calibradas. Entonces, las matrices $\mathtt{K}_1$ y $\mathtt{K}_2$ que contienen los parámetros intrínsecos de las dos cámaras son necesarios para transformar las coordenadas normalizadas en coordenadas de píxel:

$$\begin{aligned} \mathbf{x}_1 &= \mathtt{K}_1\bar{\mathbf{x}}_1 \\ \mathbf{x}_2 &= \mathtt{K}_2\bar{\mathbf{x}}_2. \end{aligned}$$

[1] Si $\mathbf{t} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$, entonces $[\mathbf{t}]_\times = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix}$.

Esto implica la siguiente ecuación:

$$\begin{aligned}(\mathtt{K}_2^{-1}\mathbf{x}_2)^\top(\mathbf{t}\times\mathtt{R}\mathtt{K}_1^{-1}\mathbf{x}_1) &= 0\\ \mathbf{x}_2^\top\mathtt{K}_2^{-\top}(\mathbf{t}\times\mathtt{R}\mathtt{K}_1^{-1}\mathbf{x}_1) &= 0 \qquad (6.1)\\ \mathbf{x}_2^\top\mathtt{F}\mathbf{x}_1 &= 0 \qquad (6.2)\end{aligned}$$

donde $\mathtt{F}=\mathtt{K}_2^{-\top}\mathtt{E}\mathtt{K}_1^{-1}$ es la *matriz fundamental*. Históricamente, la matriz esencial es anterior a la matriz fundamental.

Las matrices esencial y fundamental describen completamente las relaciones geométricas entre puntos correspondientes de un par de cámaras estéreo. La única diferencia entre las dos es que la primera se usa con cámaras calibradas, mientras que la segunda se utiliza en caso de cámaras no calibradas.

La matriz esencial contiene cinco parámetros (tres que definen la rotación y dos para la dirección de traslación - la magnitud de la traslación no es recuperable) y tiene dos restricciones: (1) su determinante vale cero, y (2) sus dos valores singulares no nulos son iguales.

La matriz fundamental contiene siete parámetros: una matriz de 9 elementos, salvo proporcionalidad y salvo la restricción de rango dos: $9-1-1=7$.

Epipolos

Como la matriz fundamental es de rango 2, tiene un núcleo $\mathbf{e}$ no nulo, tal que $\mathtt{F}\mathbf{e}_1=\mathbf{0}$. El vector de coordenadas homogéneas $\mathbf{e}_1$ que expande el núcleo recibe el nombre de epipolo en la primera imagen. Geométricamente, el epipolo es el punto de la primera imagen en el que se proyecta el centro óptico de la segunda cámara, $\mathbf{e}_1=\mathtt{P}_1\mathbf{C}_2$, con $\mathtt{P}_2\mathbf{C}_2=\mathbf{0}$.

De forma análoga se define el epipolo en la segunda imagen: $\mathbf{e}_2$ es la proyección del centro óptico de la primera cámara en la segunda imagen, $\mathbf{e}_2=\mathtt{P}_2\mathbf{C}_1$, con $\mathtt{P}_1\mathbf{C}_1=\mathbf{0}$. El vector $\mathbf{e}_2$ es el núcleo por la izquierda de la matriz fundamental, $\mathbf{e}_2^\top\mathtt{F}=\mathbf{0}^\top\Leftrightarrow\mathtt{F}^\top\mathbf{e}_2=\mathbf{0}$.

Rectas epipolares

Las rectas epipolares son las intersecciones del plano epipolar con los planos de las imágenes, según se aprecia en la figura 6.2. Para cada punto en la segunda imagen $\mathbf{x}_2$, existe una recta epipolar correspondiente $\mathbf{l}_1$ en la primera imagen. Cualquier punto $\mathbf{x}_1$ que sea el correspondiente de $\mathbf{x}_2$ debe estar sobre la recta epipolar $\mathbf{l}_1$. Esta recta epipolar también se puede ver como la proyección del rayo reproyectado que une el centro óptico de la segunda cámara $\mathbf{C}_2$ con el punto proyectado $\mathbf{x}_2$.

Así que existe una aplicación entre puntos en una imagen y sus respectivas rectas epipolares en las otra imagen. $\mathbf{x}_1\mapsto\mathbf{l}_2$ y $\mathbf{x}_2\mapsto\mathbf{l}_1$. La matriz fundamental es la que define esta correspondencia, que no es punto a punto como una homografía, sino punto a recta.

$$\begin{aligned}\mathbf{l}_2 &= \mathtt{F}\mathbf{x}_1\\ \mathbf{l}_1 &= \mathtt{F}^\top\mathbf{x}_2\end{aligned}$$

Desde el punto de vista de rectas epipolares, la ecuación de la matriz fundamental significa que cualquier punto $\mathbf{x}_2$ que sea el correspondiente de $\mathbf{x}_1$ debe estar sobre la recta epipolar $\mathbf{l}_2$, y viceversa. El punto $\mathbf{x}_2$ está sobre la recta $\mathbf{l}_2$ sii

$$\mathbf{x}_2^\top\mathbf{l}_2=\mathbf{x}_2^\top\mathtt{F}\mathbf{x}_1=0$$

El punto $\mathbf{x}_1$ está sobre la recta $\mathbf{l}_1$ sii

$$\mathbf{l}_1^\top\mathbf{x}_1=(\mathtt{F}^\top\mathbf{x}_2)^\top\mathbf{x}_1=\mathbf{x}_2^\top\mathtt{F}\mathbf{x}_1=0$$

Derivación alternativa: Algebraica

Seguiremos el desarrollo dado en [37]. Recordemos que un punto $\mathbf{X}$ produce una imagen $\mathbf{x}$ mediante la ecuación $\mathbf{x} = \mathtt{P}\mathbf{X}$. Sin pérdida de generalidad, podemos suponer que $\mathbf{X}$ está dado respecto del sistema de coordenadas de la primera cámara para denotar las siguientes ecuaciones:

$$\begin{aligned} \lambda_1 \mathbf{x}_1 &= \mathtt{K}_1[\mathbf{I}\ \mathbf{0}]\mathbf{X} \\ \lambda_2 \mathbf{x}_2 &= \mathtt{K}_2[\mathtt{R}\ \mathbf{t}]\mathbf{X} \end{aligned}$$

donde λ_1 y λ_2 son factores de escala, $\mathbf{I}$ es la matriz identidad de 3×3 y $\mathbf{0}$ es el vector nulo de 3×1. Poniendo $\mathbf{X} = [\hat{\mathbf{X}}^\top\ 1]^\top$ ($\hat{\mathbf{X}}$ es de tamaño 3×1), llegamos a la siguiente relación:

$$\begin{aligned} \lambda_2 \mathbf{x}_2 &= \mathtt{K}_2[\mathtt{R}\ \mathbf{t}]\mathbf{X} \\ &= \mathtt{K}_2(\mathtt{R}\hat{\mathbf{X}}^\top + \mathbf{t}) \\ &= \lambda_1 \mathtt{K}_2 \mathtt{R} \mathtt{K}_1^{-1} \mathbf{x}_1 + \mathtt{K}_2 \mathbf{t} \qquad (6.3) \\ \lambda_2 \mathtt{K}_2^{-1} \mathbf{x}_2 &= \lambda_1 \mathtt{R} \mathtt{K}_1^{-1} \mathbf{x}_1 + \mathbf{t}. \qquad (6.4) \end{aligned}$$

Geométricamente, esta ecuación dice que el vector de la izquierda es una combinación lineal de los dos vectores de la derecha. Por lo tanto, son coplanarios, y el vector $\mathbf{v} = \mathbf{t} \times \mathtt{R}\mathtt{K}_1^{-1}\mathbf{x}_1$ es perpendicular a ese plano:

$$\begin{aligned} \lambda_2 (\mathtt{K}_2^{-1}\mathbf{x}_2)^\top \mathbf{v} &= \lambda_1 (\mathtt{R}\mathtt{K}_1^{-1}\mathbf{x}_1)^\top \mathbf{v} + \mathbf{t}^\top \mathbf{v} = 0 \\ \mathbf{x}_2^\top (\mathtt{K}_2^{-\top}(\mathbf{t} \times \mathtt{R})\mathtt{K}_1^{-1}\mathbf{x}_1) &= 0, \end{aligned}$$

que es idéntica a (6.2). Del mismo modo, el vector $\mathbf{w} = \mathtt{K}_2\mathbf{t} \times \mathtt{K}_2\mathtt{R}\mathtt{K}_1^{-1}\mathbf{x}_1$ es perpendicular a los vectores en (6.4) premultiplicados por $\mathtt{K}_2$:

$$\begin{aligned} \lambda_2 \mathbf{x}_2^\top \mathbf{w} &= \lambda_1 \mathtt{K}_2 (\mathtt{R}\mathtt{K}_1^{-1}\mathbf{x}_1)^\top \mathbf{w} + (\mathtt{K}_2\mathbf{t})^\top \mathbf{w} = 0 \\ \mathbf{x}_2^\top (\mathtt{K}_2\mathbf{t} \times \mathtt{K}_2\mathtt{R}\mathtt{K}_1^{-1}\mathbf{x}_1) &= 0. \end{aligned}$$

Este es un resultado sorprendente porque nos da una nueva expresión equivalente para $\mathtt{F}$:

$$\mathtt{F} = [\mathtt{K}_2\mathbf{t}]_\times \mathtt{K}_2\mathtt{R}\mathtt{K}_1^{-1}, \qquad (6.5)$$

que indica que $\mathtt{F}$ se puede escribir como el producto de una matriz antisimétrica $[\mathtt{K}_2\mathbf{t}]_\times$ por una matriz invertible $\mathtt{K}_2\mathtt{R}\mathtt{K}_1^{-1}$.

Como referencia, resumimos las ecuaciones más relevantes para las matrices esencial y fundamental:

$$\begin{aligned} \mathtt{E} &= [\mathbf{t}]_\times \mathtt{R} \\ \mathtt{F} &= \mathtt{K}_2^{-\top}\mathtt{E}\mathtt{K}_1^{-1} \\ &= [\mathtt{K}_2\mathbf{t}]_\times \mathtt{K}_2\mathtt{R}\mathtt{K}_1^{-1} \end{aligned}$$

6.2.2. Planteamiento del problema de estimación

Enunciamos el problema de estimación de la matriz fundamental. Para seguir la notación de [1], todo lo referente a la segunda imagen lleva una prima.

ENUNCIADO: Dadas n parejas de puntos correspondientes, $\{(\mathbf{x}_i, \mathbf{x}'_i) | i = 1, \ldots, n\}$, estimar la matriz fundamental, $\mathtt{F}$. Veremos que es necesario un número suficientemente grande de parejas de puntos: $n \geq 7$.

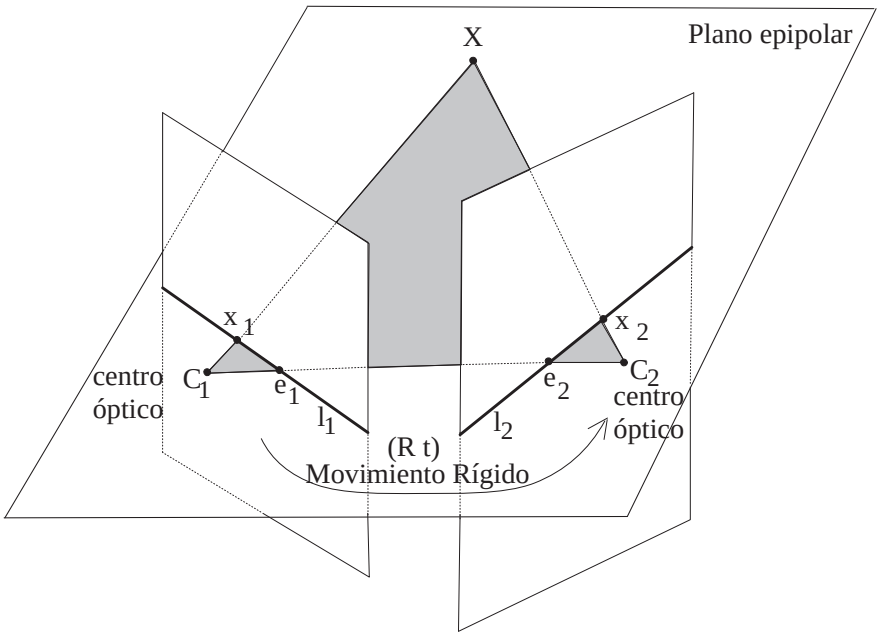


Figura 6.2: La geometría epipolar.

Con los datos $\{(\mathbf{x}_i, \mathbf{x}'_i)\}$ sabemos que existe una matriz $\mathtt{F}$ de 3×3 y de rango 2 (singular $\Leftrightarrow$ determinante nulo) que cumple

$$\mathbf{x}'^{\top}_i \mathtt{F} \mathbf{x}_i = 0$$

Admitimos que una parte de las parejas sean incorrectas: tengan errores o ruido añadido. También se ha alcanzado el objetivo de buscar técnicas de estimación robustas como RANSAC para paliar el efecto de los errores (§ 6.2.3.7).

6.2.3. Técnicas de estimación de la matriz fundamental

Sea un punto $\widetilde{\mathbf{x}}_i = [u_i, v_i]^\top$ en la primera imagen, al que se le hace corresponder el punto $\widetilde{\mathbf{x}}'_i = [u'_i, v'_i]^\top$ en la segunda imagen. Ambos deben satisfacer la restricción epipolar $\mathbf{x}'^{\top}_i \mathtt{F} \mathbf{x}_i = 0$. Esta ecuación se puede escribir como una ecuación lineal y homogénea en los nueve coeficientes desconocidos de la matriz $\mathtt{F}$:

$$\mathbf{u}_i^\top \mathbf{f} = 0, \tag{6.6}$$

donde

$$\begin{aligned} \mathbf{u}_i &= [u'_i u_i, u'_i v_i, u'_i, v'_i u_i, v'_i v_i, v'_i, u_i, v_i, 1]^\top \\ \mathbf{f} &= [\mathtt{F}_{11}, \mathtt{F}_{12}, \mathtt{F}_{13}, \mathtt{F}_{21}, \mathtt{F}_{22}, \mathtt{F}_{23}, \mathtt{F}_{31}, \mathtt{F}_{32}, \mathtt{F}_{33}]^\top, \end{aligned}$$

donde $\mathtt{F}_{ij}$ es el elemento de $\mathtt{F}$ en la fila i y la columna j.
Si nos dan n parejas de puntos, lo podemos modelar por el siguiente sistema a resolver:

$$\mathbf{U}_n \mathbf{f} = \mathbf{0},$$

donde

$$\mathbf{U}_n = [\mathbf{u}_1 \ \ \ldots \ \ \mathbf{u}_n]^\top.$$

Este conjunto de ecuaciones lineales homogéneas, junto con la restricción de rango dos de la matriz $\mathtt{F}$, nos permite estimar la geometría epipolar .

6.2.3.1. Solución exacta con 7 parejas de puntos

La matriz fundamental $\mathtt{F}$ tiene sólo 7 grados de libertad. Así que 7 es el mínimo número de parejas de puntos requeridos para obtener una solución de la geometría epipolar.

En este caso, $n = 7$ y $\operatorname{rank} \mathbf{U}_7 = 7$. Mediante la descomposición en valores singulares (SVD), podemos obtener los vectores $\mathbf{f}_1$ y $\mathbf{f}_2$ que generan (son base) el núcleo de $\mathbf{U}_7$, que corresponde a las matrices $\mathtt{F}_1$ $\mathtt{F}_2$, respectivamente. Debido a la homogeneidad, la matriz fundamental es una de la familia de matrices uniparamétricas $\alpha\mathtt{F}_1 + (1-\alpha)\mathtt{F}_2$. Como el determinante de $\mathtt{F}$ debe ser cero:

$$\det[\alpha\mathtt{F}_1 + (1-\alpha)\mathtt{F}_2] = 0, \tag{6.7}$$

de esta ecuación obtenemos un polinomio cúbico en α. El máximo número de soluciones reales es 3. Para cada solución, la matriz fundamental es:

$$\mathtt{F} = \alpha\mathtt{F}_1 + (1-\alpha)\mathtt{F}_2$$

Este algoritmo se utiliza como parte del algoritmo robusto mediante RANSAC § 6.2.3.7. Más referencias: § 10.1.2 de [1, pág. 264]. También ha sido utilizado para estimar la matriz esencial cuando los datos son 7 parejas de puntos en coordenadas normalizadas.

6.2.3.2. Método analítico con 8 o más parejas de puntos

En la práctica, nos dan más de 7 parejas de puntos. Si ignoramos la restricción de rango 2 (singularidad) de la matriz $\mathtt{F}$, podemos utilizar un método basado en mínimos cuadrados para minimizar la siguiente función de coste, a veces llamada *distancia algebraica* o *criterio lineal*:

$$\min_{\mathtt{F}} \sum_i (\mathbf{x}_i'^{\top} \mathtt{F} \mathbf{x}_i)^2, \tag{6.8}$$

que se puede escribir como:

$$\min_{\mathtt{F}} \sum_i (\mathbf{x}_i'^{\top} \mathtt{F} \mathbf{x}_i)^2 = \min_{\mathtt{F}} \sum_i (\mathbf{u}_i^{\top} \mathbf{f})^2 = \min_{\mathbf{f}} \|\mathbf{U}_n \mathbf{f}\|^2.$$

El vector $\mathbf{f}$ está definido a falta de un factor de escala. La solución trivial del problema es $\mathbf{f} = \mathbf{0}$, que no es útil. Para evitarla, necesitamos imponer alguna restricción en los coeficientes de la matriz fundamental. Hay varios métodos posibles, los llamados "algoritmos de 8 puntos", aunque se pueden utilizar más de 8 parejas. Sólo desarrollaremos uno, cuya descripción se puede encontrar en las citas: [28], Capítulo 7, pág 156; [1], Capitulo 10.1 pág 262; [30], apartado 3.2.

Análisis de autovalores
Este método consiste en imponer una restricción en la norma de $\mathbf{f}$, y en particular, se suele poner $\|\mathbf{f}\| = 1$. Ningún coeficiente de $\mathtt{F}$ prevalece en importancia sobre el resto. Volvemos al problema clásico que se expone en el apéndice B.3.2, mas esta vez debido a errores en las medidas de los puntos en las imágenes no es seguro que la matriz de diseño, $\mathbf{U}_n$, sea degenerada.

$$\min_{\mathbf{f}} \|\mathbf{U}_n \mathbf{f}\|^2 \quad \text{sujeto a} \quad \|\mathbf{f}\| = 1$$

Fundamentos de la solución
El problema se puede transformar en uno de optimización sin restricciones a través de los multiplicadores de Lagrange:

$$\min_{\mathbf{f}} \mathcal{F}(\mathbf{f}, \lambda),$$

donde

$$\mathcal{F}(\mathbf{f}, \lambda) = \|\mathbf{U}_n \mathbf{f}\|^2 + \lambda(1 - \|\mathbf{f}\|^2)$$

y λ es el multiplicador de Lagrange. Al igualar la primera derivada de $\mathcal{F}(\mathbf{f}, \lambda)$ con respecto a $\mathbf{f}$ (gradiente) a cero, resulta:

$$\mathbf{U}_n^\top \mathbf{U}_n \mathbf{f} = \lambda \mathbf{f}.$$

Así pues, la solución $\mathbf{f}$ debe ser un autovector unitario de la matriz $\mathbf{U}_n^\top \mathbf{U}_n$ de 9×9 y λ es el autovalor correspondiente. Como la matriz $\mathbf{U}_n^\top \mathbf{U}_n$ es simétrica y semidefinida positiva, todos sus autovalores son reales positivos o cero. Sin pérdida de generalidad, suponemos que los nueve autovalores de $\mathbf{U}_n^\top \mathbf{U}_n$ están en orden descendente:

$$\lambda_1 \geq \cdots \geq \lambda_i \geq \cdots \geq \lambda_9 \geq 0.$$

Por lo tanto, tenemos 9 posibles soluciones: $\lambda = \lambda_i$ para $i = 1, \ldots, 9$. Sustituyendo cualquiera de las soluciones en la función $\mathcal{F}$ se obtiene:

$$\mathcal{F}(\mathbf{f}, \lambda_i) = \lambda_i.$$

Como queremos minimizar $\mathcal{F}(\mathbf{f}, \lambda)$, la solución del problema es el autovector de $\mathbf{U}_n^\top \mathbf{U}_n$ asociado al menor autovalor, esto es, λ_9. El coste del ajuste es $\|\mathbf{U}_n \mathbf{f}\|^2 = \lambda_9$, aunque se suele tomar la norma, no su cuadrado, luego coste $= \|\mathbf{U}_n \mathbf{f}\| = \sqrt{\lambda_9}$.

Relación con la SVD

La descomposición en autovalores y autovectores de $\mathbf{U}_n^\top \mathbf{U}_n$ es:

$$\mathbf{U}_n^\top \mathbf{U}_n = \mathtt{V} \mathbf{\Lambda} \mathtt{V}^{-1} = \mathtt{V} \mathbf{\Lambda} \mathtt{V}^\top$$

ya que la matriz de autovectores es ortonormal: $\mathtt{V}^{-1} = \mathtt{V}^\top$. Por otra parte, si $\mathbf{U}_n = \mathtt{UDV}^\top$ es la SVD de $\mathbf{U}_n$, tenemos que

$$\mathbf{U}_n^\top \mathbf{U}_n = \mathtt{VDU}^\top \mathtt{UDV}^\top = \mathtt{VD}^2 \mathtt{V}^\top$$

Es decir, $\mathtt{D}^2 = \mathbf{\Lambda}$ y la matriz de autovectores coincide con la matriz $\mathtt{V}$ de vectores singulares. Así que el autovector de $\mathbf{U}_n^\top \mathbf{U}_n$ asociado al menor autovalor, λ_9, es el vector singular de $\mathtt{V}$ en la SVD $\mathbf{U}_n = \mathtt{UDV}^\top$ asociado al menor valor singular: $\sigma_9^2 = \lambda_9$, y el coste del ajuste es el menor valor singular, coste $= \sigma_9$.

6.2.3.3. Imposición de la condición de rango 2 o singularidad

La ventaja del anterior algoritmo lineal es que produce una solución en forma cerrada. Aún así, es bastante sensible al ruido, incluso con un gran número de puntos dados. Una razón es que no se satisface la restricción de rango 2 ($\det(\mathtt{F}) = 0$). Podemos imponer esta condición a posteriori.

El problema que surge es: dada una aplicación $\mathtt{F}$ de un espacio euclídeo en sí mismo

$$\mathtt{F} : (\mathbb{R}^n, \|\cdot\|_2) \longrightarrow (\mathbb{R}^n, \|\cdot\|_2)$$

y definiendo la norma inducida de una matriz $\mathtt{A}$ como

$$\|\mathtt{A}\|_2 = \sup_{\|\mathbf{x}\|_2 = 1} \|\mathtt{A}\mathbf{x}\|_2$$

calcular la matriz aproximación $\hat{\mathtt{F}}$ que verifica que su determinante vale cero ($\det(\hat{\mathtt{F}}) = 0$) y que minimiza $\|\mathtt{F} - \hat{\mathtt{F}}\|_2$.

Existe un teorema que da la solución a este problema y se basa en la Descomposición en Valores Singulares de $\mathtt{F}$:

$$\mathtt{F} = \mathtt{U} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix} \mathtt{V}^\top$$

cumpliéndose que las matrices $\mathtt{U}$ y $\mathtt{V}$ son unitarias (ortogonales por tener elementos reales) y que $\sigma_1 \geq \ldots \sigma_n \geq 0$. La matriz $\hat{\mathtt{F}}$ óptima se construye como:

$$\hat{\mathtt{F}} = \mathtt{U} \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_{n-1} & \\ & & & 0 \end{pmatrix} \mathtt{V}^\top$$

y verifica

$$\|\mathtt{F}-\hat{\mathtt{F}}\|_2 = \left\|\mathtt{U}\begin{pmatrix}\sigma_1 & & \\ & \ddots & \\ & & \sigma_{n-1}\ \sigma_n\end{pmatrix}\mathtt{V}^\top - \mathtt{U}\begin{pmatrix}\sigma_1 & & \\ & \ddots & \\ & & \sigma_{n-1}\ 0\end{pmatrix}\mathtt{V}^\top\right\|_2$$

$$= \left\|\mathtt{U}\begin{pmatrix}0 & & \\ & \ddots & \\ & & 0\ \sigma_n\end{pmatrix}\mathtt{V}^\top\right\|_2 = \left\|\begin{pmatrix}0 & & \\ & \ddots & \\ & & 0\ \sigma_n\end{pmatrix}\right\|_2 = \sigma_n$$

El valor singular σ_n es la distancia mínima. Si tomamos cualquier matriz singular $\widetilde{\mathtt{F}}$, se puede demostrar que la distancia a la matriz $\mathtt{F}$ es mayor: $\|\mathtt{F}-\widetilde{\mathtt{F}}\|_2 \geq \sigma_n$. Para ello utilicemos un vector unitario $\mathbf{v}$ del núcleo de $\widetilde{\mathtt{F}}$, es decir, $\widetilde{\mathtt{F}}\mathbf{v}=\mathbf{0}$. Entonces

$$\|\mathtt{F}-\widetilde{\mathtt{F}}\|_2 \geq \|(\mathtt{F}-\widetilde{\mathtt{F}})\mathbf{v}\|_2 = \|\mathtt{F}\mathbf{v}-\widetilde{\mathtt{F}}\mathbf{v}\|_2 = \|\mathtt{F}\mathbf{v}\|_2 \geq \sigma_n = \|\mathtt{F}-\hat{\mathtt{F}}\|_2$$

Particularizando para nuestro problema concreto, $\mathtt{F}$ de dimensión 3, reemplazamos la matriz $\mathtt{F}$ estimada por cualquiera de los métodos anteriores por la matriz $\hat{\mathtt{F}}$. Sea

$$\mathtt{F} = \mathtt{U}\begin{pmatrix}\sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3\end{pmatrix}\mathtt{V}^\top$$

la SVD de la matriz $\mathtt{F}$, entonces la matriz buscada es

$$\hat{\mathtt{F}} = \mathtt{U}\begin{pmatrix}\sigma_1 & & \\ & \sigma_2 & \\ & & 0\end{pmatrix}\mathtt{V}^\top$$

Una interpretación sencilla es que estamos proyectando la matriz $\mathtt{F}$ estimada sobre el conjunto de las matrices de 3×3 singulares. Esto se deriva de las propiedades de la SVD.

6.2.3.4. Interpretación geométrica del criterio lineal

Otro problema del algoritmo lineal es que la magnitud que estamos minimizando (la llamada distancia algebraica) no tiene significado físico. Una magnitud físicamente significativa sería algo medido en el plano de la imagen, ya que ese es el origen de las observaciones y dispone de una métrica con sentido físico. Una magnitud tal es la distancia (en una imagen) desde un punto $\widetilde{\mathbf{x}}'_i = [u'_i, v'_i]^\top$ a su recta epipolar correspondiente $\mathbf{l}'_i = \mathtt{F}\mathbf{x}_i \equiv [l'_{i1}, l'_{i2}, l'_{i3}]^T$, que viene dada por:

$$d(\widetilde{\mathbf{x}}'_i, \mathbf{l}'_i) = \frac{\mathbf{x}_i'^\top \mathbf{l}'_i}{\sqrt{l'^2_{i1}+l'^2_{i2}}} = \frac{1}{c_i}\mathbf{x}_i'^\top \mathtt{F}\mathbf{x}_i, \tag{6.9}$$

donde $c_i = \sqrt{l'^2_{i1}+l'^2_{i2}}$. Así, el criterio de la ecuación (6.8) se puede escribir como:

$$\min_{\mathtt{F}} \sum_{i=1}^{n} c_i^2 d^2(\widetilde{\mathbf{x}}'_i, \mathbf{l}'_i).$$

Esto significa que en el algoritmo lineal se minimiza no sólo una cantidad física $d(\widetilde{\mathbf{x}}'_i, \mathbf{l}'_i)$, sino también c_i que no es físicamente medible [30].

Seguidamente se presentan algunos algoritmos iterativos no lineales que buscan la matriz fundamental dentro de la variedad formada por las matrices singulares.

6.2.3.5. Minimización de la distancia algebraica

El algoritmo lineal de los 8 puntos ya minimiza la distancia algebraica, pero no dentro de la variedad de las matrices singulares, sino en todo el espacio ambiente. La restricción de singularidad se impone una vez hallada la solución.

El presente algoritmo calcula la matriz fundamental utilizando la propiedad de que dicha matriz se puede descomponer como el producto de una matriz antisimétrica dada por su epipolo por una matriz regular. Nos referimos al algoritmo explicado del artículo [3], § 3. Computation of the Fundamental Matrix, (b) Algebraic Minimization. Es el algoritmo 10.2 de [1, pág. 267]: "Computation of $\mathtt{F}$ with $\det(\mathtt{F}) = 0$ by iteratively minimizing algebraic error", aunque está mejor comentado en el artículo.

El enunciado del problema a resolver es el siguiente:

$$\min_{\mathtt{F}} \sum_i (\mathbf{x}_i'^{\top} \mathtt{F} \mathbf{x}_i)^2 = \min_{\mathbf{f}} \|\mathtt{M}\hat{\mathbf{f}}\|^2 \quad \text{sujeto a} \quad \|\hat{\mathbf{f}}\| = 1 \ \text{ y } \det \hat{\mathbf{f}} = 0$$

Funciones modelo y de coste
Siguiendo el artículo, la función modelo utilizada en el contexto de los algoritmos Levenberg-Marquardt para este problema concreto es:

$$\begin{array}{rccc} f: & \mathbb{R}^3 & \longrightarrow & \mathbb{R}^9 \\ & \mathbf{e} & \mapsto & \epsilon = \mathtt{ME}\mathbf{q} = \mathtt{M}\hat{\mathbf{f}} \end{array}$$

En esta parametrización, el vector de parámetros es el epipolo (proyección del centro óptico de la segunda cámara en la primera imagen) y una vez conocido, es posible calcular una matriz fundamental consistente con el mismo, de tal forma que aplicada a la matriz de diseño $\mathtt{M}$, obtenida a partir de las observaciones en las imágenes, proporciona un vector de error ϵ cuya norma es la distancia algebraica.

$$\|\epsilon\| = \|\mathtt{M}\hat{\mathbf{f}}\| = \|\mathtt{ME}\mathbf{q}\|$$

En este caso, el vector de medidas $\widehat{\mathbf{X}}$, tiene dimensión $N = 9$. El vector $\mathbf{X}$ objetivo es aquel para el cual el coste es cero, es decir, el vector nulo, como se explica en D.1.3.

La matriz de restricción $\mathtt{E}$, de 9×9, se construye a partir del epipolo, $\mathtt{E} = \mathrm{diag}([\mathbf{e}]_\times, [\mathbf{e}]_\times, [\mathbf{e}]_\times)$. Esta matriz verifica $\mathrm{rank}(\mathtt{E}) = 6$ y sirve para imponer la restricción de singularidad de la matriz fundamental. En $\mathbb{R}^9$, limitamos la búsqueda de la matriz fundamental al subespacio generado por las columnas de $\mathtt{E}$.

$$\min_{\mathbf{q}} \|\mathtt{ME}\mathbf{q}\| \ \text{ con la restriccion } \|\hat{\mathbf{f}}\| = \|\mathtt{E}\mathbf{q}\| = 1$$

Este problema se resuelve mediante el algoritmo A.3.7, § A3.4.4 de [1], que utiliza la SVD. El epipolo desde el cual iniciar la búsqueda es obtenido a partir de una estimación inicial de la matriz fundamental dada por el algoritmo lineal de los 8 puntos.

La función de coste calcula la distancia algebraica, que es la norma del vector de error o residuo, ϵ.

$$\text{coste} = \|\epsilon\| = \|\mathtt{ME}\mathbf{q}\|$$

En toda la minimización, se mantiene la matriz de diseño obtenida a partir de los datos normalizados para mejorar la estabilidad numérica. Es incluso preferible substituir $\mathtt{M}$ de $n \times 9$, donde n es el número de parejas de puntos, por la matriz de diseño/medida reducida correspondiente, $\hat{\mathtt{M}}$ cuadrada de dimensión 9, para evitar realizar más operaciones de las necesarias.

El algoritmo de minimización de la distancia algebraica es mejor que el algoritmo de los ocho puntos, aunque se utiliza poco en este proyecto, ya que está implementado el Gold Standard.

6.2.3.6. Minimización del error de reproyección: Gold Standard

En la mayoría de los casos, se asume un modelo de ruido gaussiano sumado a las coordenadas de los puntos exactos en las imágenes, tal como se dice en § 10.4.1 de [1, pág. 268]. Bajo esta hipótesis, la mejor estimación consiste en minimizar la distancia geométrica medida en las imágenes (el error de reproyección)

$$\sum_i \left[d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 \right], \tag{6.10}$$

donde $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ son parejas medidas, mientras que $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$ son verdaderas correspondencias estimadas que satisfacen $\hat{\mathbf{x}}_i'^{\top} \mathtt{F} \hat{\mathbf{x}}_i = 0$ para cierta matriz $\mathtt{F}$ de rango 2.

Por lo tanto, el problema estriba en encontrar la matriz fundamental $\mathtt{F}$ y las parejas de puntos corregidos $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$. Esto es lo que realiza el algoritmo llamado "Gold Standard", un algoritmo que requiere conocimientos adicionales a los ya expuestos. Por ejemplo, utiliza la triangulación y la calibración proyectiva de dos cámaras, que se explicarán más adelante.

A pesar de ello, adelantamos que la matriz fundamental se parametriza mediante dos matrices de proyección compatibles con ella, como se indica en § 6.2.5.2 y que parte de los parámetros respecto de los que es optimizada la función de coste son los puntos 3D de los que proceden las proyecciones. La figura 6.3 recoge el planteamiento del algoritmo para un punto: las observaciones en las dos imágenes $\mathbf{x}, \mathbf{x}'$, el punto 3D estimado del que proceden $\hat{\mathbf{X}}$ y las proyecciones de este punto, $\hat{\mathbf{x}}, \hat{\mathbf{x}}'$, que son los puntos corregidos que forman una verdadera correspondencia ya que proceden del mismo punto 3D por construcción.

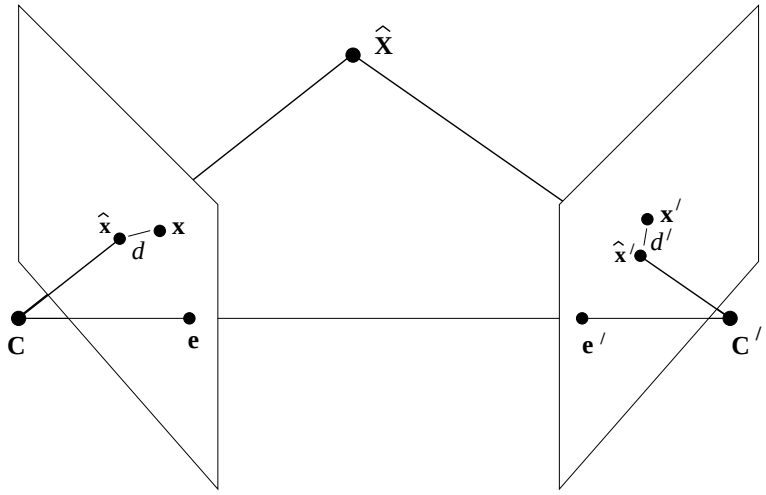


Figura 6.3: Minimización del error de reproyección

Funciones modelo y de coste
Según § A4.5 de [1, pág. 578], la función modelo a través de la cual se minimiza la función de coste (6.10) es:

$$f : (\mathtt{P}', \mathbf{b}_1, \ldots, \mathbf{b}_n) \equiv \mathbf{P} \rightarrow \hat{\mathbf{X}} \equiv (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}'_1, \ldots, \hat{\mathbf{x}}_n, \hat{\mathbf{x}}'_n)$$

Se considera que la matriz de proyección de la primera cámara es la canónica, $[\mathtt{I} \mid \mathbf{0}]$. El vector de parámetros $\mathbf{P}$ contiene en sus $M = 3n + 12$ componentes la información de la matriz de proyección de la segunda cámara y las coordenadas afines de los puntos 3D a estimar.

$$\mathbf{b}_i = (X_i, Y_i, T_i)^{\top} \quad ,, \quad \hat{\mathbf{X}}_i = (X_i, Y_i, 1, T_i)^{\top}$$

Para calcular una estimación inicial de los puntos 3D $\hat{\mathbf{X}}_i$ es necesaria una reconstrucción mediante triangulación, por ejemplo la triangulación óptima explicada en [1, pág. 304].

El vector de medidas $\widehat{\mathbf{X}}$, de dimensión $N = 4n$ posee las coordenadas afines de los puntos proyectados estimados. No confundir éste último $\widehat{\mathbf{X}}$ y sus elementos $\widehat{\mathbf{X}}_i$ con las coordenadas $\mathbf{b}_i$ de los puntos 3D $\hat{\mathbf{X}}_i$.

La función de coste se expresa en términos de la función modelo:

$$\text{coste} = \|\mathbf{X} - f(\mathbf{P})\| = \sqrt{\sum_i \left[d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2\right]}$$

La opción de minimizar la función de coste directamente no es numéricamente recomendable, es mejor explotar el conocimiento del modelo. Debido a este conocimiento, las submatrices $\mathtt{A}_i(4\times 12)$ y $\mathtt{B}_i(4\times 3)$ de la matriz jacobiana tienen una forma conocida, por lo que numéricamente es más rápido construirla de forma exacta a partir de los datos disponibles que estimarla.

Suponiendo que un punto $\mathbf{b}_i$ tiene por coordenadas homogéneas $\hat{\mathbf{X}}_i$ y que $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, las coordenadas homogéneas de los puntos proyectados son: $\hat{\mathbf{x}}_{ih} = \mathtt{P}\hat{\mathbf{X}}_i = (X_i, Y_i, 1)^\top$ y $\hat{\mathbf{x}}'_{ih} = \mathtt{P}'\hat{\mathbf{X}}_i$. Las coordenadas afines que componen $\widehat{\mathbf{X}}$ son:

$$\widehat{\mathbf{X}}_i = \begin{pmatrix} \hat{\mathbf{x}}_i \\ \hat{\mathbf{x}}'_i \end{pmatrix} \quad ,, \quad \hat{\mathbf{x}}_i = \left(\frac{x_i}{w_i}, \frac{y_i}{w_i}\right)^\top = (X_i, Y_i)^\top \quad ,, \quad \hat{\mathbf{x}}'_i = \left(\frac{x'_i}{w'_i}, \frac{y'_i}{w'_i}\right)^\top$$

Y las matrices jacobianas:

$$\mathtt{A}_i = \frac{\partial \widehat{\mathbf{X}}_i}{\partial \mathbf{a}} = \begin{bmatrix} \mathbf{0} \\ \dfrac{\partial \hat{\mathbf{x}}'_i}{\partial \mathbf{a}} \end{bmatrix} \quad ,, \quad \frac{\partial \hat{\mathbf{x}}'_i}{\partial \mathbf{a}} = \frac{1}{w'_i} \begin{bmatrix} \hat{\mathbf{X}}_i^\top & \mathbf{0} & -\dfrac{x'_i}{w'_i}\hat{\mathbf{X}}_i^\top \\ \mathbf{0} & \hat{\mathbf{X}}_i^\top & -\dfrac{y'_i}{w'_i}\hat{\mathbf{X}}_i^\top \end{bmatrix} = \begin{bmatrix} \mathtt{I} & -\hat{\mathbf{x}}'_i \end{bmatrix} \otimes \frac{1}{w'_i}\hat{\mathbf{X}}_i^\top$$

$$\mathtt{B}_i = \frac{\partial \widehat{\mathbf{X}}_i}{\partial \mathbf{b}_i} = \begin{bmatrix} \mathtt{I} \mid \mathbf{0} \\ \dfrac{\partial \hat{\mathbf{x}}'_i}{\partial \mathbf{b}_i} \end{bmatrix} \quad ,, \quad \frac{\partial \hat{\mathbf{x}}'_i}{\partial \mathbf{b}_i} = \frac{1}{w'_i}\left(\begin{bmatrix} \mathtt{P}_{11} & \mathtt{P}_{12} & \mathtt{P}_{14} \\ \mathtt{P}_{21} & \mathtt{P}_{22} & \mathtt{P}_{24} \end{bmatrix} - \hat{\mathbf{x}}'_i[\mathtt{P}_{31}\ \mathtt{P}_{32}\ \mathtt{P}_{34}]\right)$$

6.2.3.7. Estimación robusta de mediante RANSAC

Se propone otro algoritmo para conocer la geometría epipolar entre dos imágenes, el cual incluye estimación robusta mediante RANSAC, siguiendo las recomendaciones de § 10.6, [1, pág. 274].

El algoritmo se puede resumir en los siguientes pasos: dadas las correspondencias de puntos entre imágenes $(\mathbf{x}_i, \mathbf{x}'_i)$, $i = 1, \ldots, n$

1. **RANSAC básico**: Repetir hasta haber probado N muestras (donde N es determinado adaptativamente según el algoritmo 3.5 de [1, pág. 105]) o haber logrado más de $T = (1 - \epsilon_{\text{fijo}})n$ puntos consistentes con el modelo (*inliers*).

   a) Seleccionar una muestra aleatoria de 7 parejas de puntos y calcular la matriz fundamental $\mathtt{F}$ (§ 6.2.3.1). Puede haber una o tres soluciones reales.

   b) Calcular la distancia $d_\perp$ de cada punto al modelo de geometría epipolar indicado por $\mathtt{F}$.

   c) Calcular el número de puntos consistentes con el modelo $\mathtt{F}$ como el número de correspondencias de puntos para los que $d_\perp < t = \sqrt{3{,}84}\sigma$ píxeles, siendo σ la desviación típica del ruido gaussiano esperado.

   d) Si hay tres soluciones reales para $\mathtt{F}$, se calcula el número de *inliers* para cada solución, y se retiene la solución con mayor número de *inliers* (mayor soporte).

e) Comparar: Elegir la $\mathtt{F}$ con el mayor soporte. En el caso de empate, retener la solución con la menor desviación típica de *inliers* (en cuanto a distancia al modelo $d_\perp$).

f) Estimar la proporción ϵ de puntos no consistentes (*outliers*) y el número de muestras aleatorias a probar, N.

2. **Estimación óptima no lineal**: re-estimar $\mathtt{F}$ minimizando la distancia geométrica - Error de reproyección ec. (6.10) - a partir del mayor y mejor soporte encontrado. Se utiliza el algoritmo Gold Standard sólo sobre las correspondencias clasificadas como consistentes con el modelo.

3. **Identificación de nuevas parejas** consistentes con el modelo, igual que en los pasos c) y e). Continuar minimizando la distancia geométrica (paso 2) hasta que el soporte deje de crecer.

De las dos opciones que se indican en [1] sobre la elección de la distancia $d_\perp$ de cada punto al modelo, se calcula el error de reproyección de forma exacta, no la aproximación de Sampson al mismo, para ser consistentes. Para ver si los puntos son consistentes con el modelo o no es necesario triangular, obtener los puntos 3D de los puntos de los que desconocemos sus proyecciones y proyectarlos.

6.2.4. Limitación de la autocalibración proyectiva

Una limitación de la autocalibración proyectiva es que la solución es única a falta de una homografía del espacio, representada por una matriz H de 4×4 regular ($\det(\mathtt{H}) \neq 0$).

Si dadas las parejas de puntos $\{(\mathbf{x}_i, \mathbf{x}'_i)\}$ conseguimos obtener unas matrices que representan las proyecciones $(\mathtt{P}, \mathtt{P}')$ con las que podemos hallar la reconstrucción de los puntos $\mathbf{X}_i$ en el espacio proyectivo $\mathbb{P}^3$, esto es:

$$(\mathbf{x}_i, \mathbf{x}'_i) \longrightarrow (\mathtt{P}, \mathtt{P}', \{\mathbf{X}_i\})$$

con

$$\begin{array}{rcl} \mathbf{x}_i & \sim & \mathtt{P}\mathbf{X}_i \\ \mathbf{x}'_i & \sim & \mathtt{P}'\mathbf{X}_i \end{array}$$

existe otra solución:

$$(\mathbf{x}_i, \mathbf{x}'_i) \longrightarrow (\mathtt{PH}, \mathtt{P}'\mathtt{H}, \{\mathtt{H}^{-1}\mathbf{X}_i\})$$

Esto es fácil de verificar:

$$\mathbf{x}_i \sim \mathtt{P}\mathbf{X}_i = \mathtt{PHH}^{-1}\mathbf{X}_i = (\mathtt{PH})(\mathtt{H}^{-1}\mathbf{X}_i) = \tilde{\mathbf{P}}\tilde{\mathbf{X}}_i$$

y lo mismo para los puntos de la segunda imagen.

El conocimiento de la matriz $\mathtt{F}$ no nos permite llegar al máximo refinamiento de la solución deseado. O lo que es lo mismo, sólo a partir de la matriz fundamental es imposible obtener la estructura euclídea del espacio. Pero antes daremos unas propiedades de la matriz fundamental.

6.2.5. Relación de $\mathtt{F}$ con las matrices de proyección

Una aplicación directa una vez conocida la matriz fundamental es la calibración proyectiva de las dos cámaras. También es posible y fácil el paso inverso: la obtención de la matriz fundamental a partir de las matrices de proyección. Veamos primero la última conversión.

6.2.5.1. Matriz fundamental de dos matrices de proyección

Dadas dos matrices de proyección $\mathtt{P}$ y $\mathtt{P}'$, la matriz fundamental $\mathtt{F}$ asociada a ambas, según el resumen de [1, pág. 226] es:

$$\mathtt{F} = [\mathbf{e}']_\times \mathtt{P}'\mathtt{P}^+$$

donde $\mathtt{P}^+$ es la pseudoinversa de $\mathtt{P}$, y $\mathbf{e}' = \mathtt{P}'\mathbf{C}$, con $\mathtt{P}\mathbf{C} = \mathbf{0}$ ($\mathbf{C}$ son las coordenadas homogéneas del centro óptico de la primera cámara). En este caso no hace falta estimar $\mathtt{F}$, ya que se construye de forma directa a partir de las matrices de proyección.

6.2.5.2. Calibración proyectiva de dos cámaras

Una forma de calcular dos matrices de proyección compatibles con una matriz fundamental $\mathtt{F}$ dada se indica en el resultado 8.14 de [1, pág. 237]. Siempre se puede elegir que la matriz de la primera cámara sea la canónica $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, entonces la matriz de la segunda cámara es $\mathtt{P}' = [[\mathbf{e}]'_\times \mathtt{F} \mid \mathbf{e}']$.

6.3. Triangulación

La triangulación consiste en el cálculo la posición de un punto 3D $\mathbf{X}_i$ del espacio dadas sus proyecciones en dos imágenes $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ y las matrices de proyección de ambas cámaras $\mathtt{P}$ y $\mathtt{P}'$. En adelante supondremos que sólo hay errores en las coordenadas de los puntos observados en las imágenes, no en las matrices de proyección.

Sigamos la ilustración de la figura 6.4. Bajo la hipótesis de existencia de ruido, los rayos reproyectados (rectas con origen en el centro óptico de cada cámara y que pasan por los correspondientes puntos en cada imagen) no se cortarán, en general, por lo que hay que estimar una mejor aproximación al punto 3D.

La mejor aproximación requiere definir una función de coste a minimizar. En las reconstrucciones proyectiva y afín no es posible definir una métrica en el espacio 3D, ya que se desconoce la estructura euclídea, así que las funciones de coste deben basarse en medidas en el plano de la imagen, como venimos haciendo. Es conveniente utilizar un método de triangulación que sea invariante ante transformaciones proyectivas del espacio.

En estas secciones se tratan dos algoritmos de triangulación: uno cuya resolución es posible por un método lineal y otro, estimador óptimo, cuya solución se obtiene sin una minimización iterativa. En este escenario, suponemos que la matriz fundamental $\mathtt{F}$ es dada *a priori* y se debe determinar la posición del punto en el espacio, $\mathbf{X}$. Llamamos n al número de parejas de puntos.

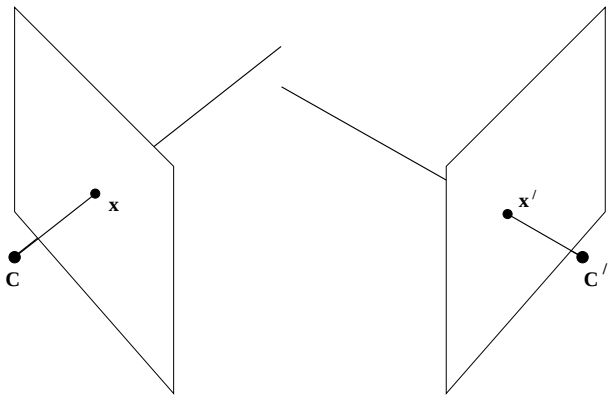


Figura 6.4: En general, rayos reproyectados de puntos con errores se cruzan en el espacio

6.3.1. Triangulación lineal

Existe un algoritmo lineal (DLT o SVD) como sencillo esquema de triangulación. Este algoritmo es análogo al de la estimación de la matriz de proyección § 5.4.1, y se seguirá la misma notación que allí se expuso.

Para cada correspondencia, en cada imagen tenemos una observación $\mathbf{x} = \mathtt{P}\mathbf{X}, \mathbf{x}' = \mathtt{P}'\mathbf{X}$, y estas ecuaciones se pueden expresar como un sistema de ecuaciones en $\mathbf{X}$. Al igual que en la estimación de la matriz de proyección, se elimina el factor de proporcionalidad mediante el producto vectorial: $\mathbf{x} \sim$

$\mathtt{P}\mathbf{X} \Leftrightarrow \mathbf{x} \times \mathtt{P}\mathbf{X} = \mathbf{0}$, lo que da lugar a tres ecuaciones de las cuales dos son linealmente independientes. Recordemos el sistema (5.4) y démosle la vuelta:

$$\begin{bmatrix} \mathbf{0}^\top & -w_i\mathbf{X}_i^\top & y_i\mathbf{X}_i^\top \\ w_i\mathbf{X}_i^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i^\top \\ -y_i\mathbf{X}_i^\top & x_i\mathbf{X}_i^\top & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0} \Leftrightarrow \begin{bmatrix} \mathbf{0}^\top & -w_i\mathbf{P}^{2\top} & y_i\mathbf{P}^{3\top} \\ w_i\mathbf{P}^{1\top} & \mathbf{0}^\top & -x_i\mathbf{P}^{3\top} \\ -y_i\mathbf{P}^{1\top} & x_i\mathbf{P}^{2\top} & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{X}_i \\ \mathbf{X}_i \\ \mathbf{X}_i \end{pmatrix} = \mathbf{0}$$

Este sistema se puede operar y simplificar

$$\begin{bmatrix} -w_i\mathbf{P}^{2\top} + y_i\mathbf{P}^{3\top} \\ w_i\mathbf{P}^{1\top} - x_i\mathbf{P}^{3\top} \\ -y_i\mathbf{P}^{1\top} + x_i\mathbf{P}^{2\top} \end{bmatrix} \mathbf{X}_i = \mathtt{A}_i\mathbf{X}_i = \mathbf{0}$$

Para la proyección en la segunda imagen se hace lo mismo y resulta un sistema $\mathtt{A}_i'\mathbf{X}_i = \mathbf{0}$. Juntando los dos sistemas es posible hallar una solución.

$$\mathtt{A}\mathbf{X}_i = \begin{bmatrix} \mathtt{A}_i \\ \mathtt{A}_i' \end{bmatrix} \mathbf{X}_i = \mathbf{0}$$

En la práctica no se utilizan las seis ecuaciones, sino sólo las dos primeras de cada sistema $\mathtt{A}_i, \mathtt{A}_i'$. Además, se deshomogeneizan las coordenadas de los puntos $\mathbf{x}_i, \mathbf{x}_i'$ dividiendo por la tercera coordenada, w_i y w_i', respectivamente.

Es indudable que se pretende minimizar la distancia algebraica. La solución, como ya se ha mostrado más de una vez, es el vector singular (derecho) de $\mathtt{A}$ correspondiente al menor valor singular y el mínimo de la función de coste es el menor valor singular.

Este algoritmo se suele utilizar con las observaciones de un par de cámaras, pero repasando la exposición, se aprecia que es muy fácil incluir las observaciones de cuantas cámaras se desee, basta con concatenar verticalmente las matrices $\mathtt{A}_i, \mathtt{A}_i', \mathtt{A}_i'' \ldots$ para formar una única matriz $\mathtt{A}$ para cada correspondencia de puntos. Pero ¡ cuidado!, debe tenerse en cuenta un detalle: todas las matrices de proyección deben estar expresadas en la misma referencia espacial.

6.3.2. Triangulación óptima

Volvamos sobre el concepto de la restricción epipolar: si los puntos proyectados verifican la restricción epipolar respecto de $\mathtt{F}$, existe un punto 3D $\mathbf{X}_i$ del que proceden los puntos proyectados, es decir, las ecuaciones de proyección

$$\mathbf{x}_i = \mathtt{P}\mathbf{X}_i, \quad \mathbf{x}_i' = \mathtt{P}'\mathbf{X}_i$$

lineales y homogéneas, tienen solución exacta: los puntos $\mathbf{X}_i$ buscados.

En observaciones de imágenes reales, la restricción epipolar se verificará sólo aproximadamente, así que es más adecuado plantear el problema como una estimación de máxima verosimilitud. Bajo la hipótesis habitual de que la perturbación en la detección de los puntos en las imágenes es ruido aditivo gaussiano isótropo independiente para cada punto y de la misma varianza, la estimación más verosímil es la que minimiza la suma de las distancias euclídeas al cuadrado conocida como error de reproyección. Buscamos las correspondencias de puntos $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}_i'$ que minimizan

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}_i', \hat{\mathbf{x}}_i')^2 \tag{6.11}$$

sujeto a la restricción epipolar $\hat{\mathbf{x}}_i'^\top \mathtt{F}\hat{\mathbf{x}}_i = 0$.

La solución a este problema de minimización se puede obtener de forma no iterativa mediante las raíces reales de un polinomio de sexto grado. La figura 6.5 ilustra el método de resolución, basado en

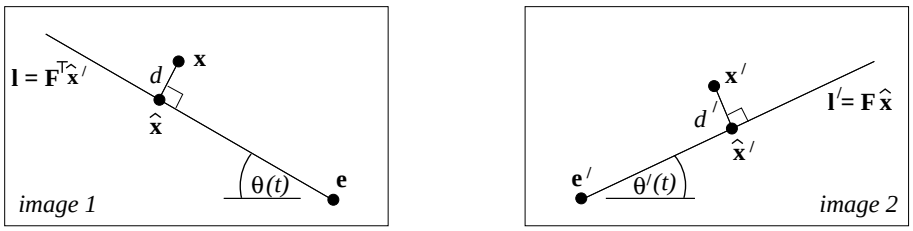


Figura 6.5: Triangulación óptima

una parametrización de las rectas epipolares correspondientes.

Los puntos buscados $\hat{\mathbf{x}}_i$, $\hat{\mathbf{x}}'_i$ son los que, estando sobre rectas epipolares correspondientes, minimizan la función de coste (6.11). Estos puntos son, precisamente, los que minimizan la distancia perpendicular de cada recta epipolar al punto observado.

$$\sum_i d(\mathbf{x}_i, \mathbf{l}_i)^2 + d(\mathbf{x}'_i, \mathbf{l}'_i)^2 \tag{6.12}$$

Falta determinar el par de rectas epipolares correspondientes que minimizan la función de coste (6.12). Para ello se parametriza el haz de rectas epipolares mediante un parámetro t y se obtienen las correspondientes rectas epipolares en la segunda imagen a través de la matriz fundamental, conocida por hipótesis. Se expresa la función de coste en función del parámetro t y utilizando reglas de cálculo elemental se minimiza la función de coste en términos de t. Si se desea profundizar en los detalles de la parametrización, etc. consúltese § 11.5.2 de [1].

Incertidumbre

La reconstrucción es más fiable cuanto mayor sea el ángulo entre los rayos reproyectados que pasan por las correspondientes proyecciones de un mismo punto 3D, como se aprecia en la figura 6.6. La zona sombreada de la figura ilustra la forma de la zona de incertidumbre, que depende del ángulo entre los rayos reproyectados. Cuanto más paralelos sean los rayos, peor es la estimación de la localización de los puntos 3D.

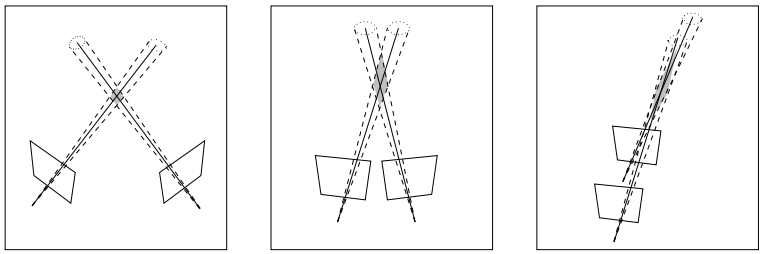


Figura 6.6: Incertidumbre en la reconstrucción

Consejo práctico: cuando se vaya a adquirir una secuencia de imágenes, si se está enfocando una escena lejana, por mucho que se desplace la cámara (*baseline*) el movimiento relativo de los puntos proyectados es pequeño porque el ángulo entre rayos reproyectados es pequeño. Hay que procurar captar imágenes desde posiciones suficientemente distantes como para que el paralaje (triangulación) sea fiable.

6.4. Geometría de tres cámaras

En esta sección se recogen los algoritmos de los capítulos 14 y 15 de [1], sobre la estimación del tensor trifocal $\mathcal{T}$ por métodos numéricos dadas unas correspondencias de puntos entre tres imágenes o proyecciones. También se trata su relación con las matrices de proyección y matrices fundamentales .

El desarrollo es muy parecido a la sección sobre la matriz fundamental, utilizando las mismas técnicas. En concreto, se trataran los siguientes algoritmos:

1. Método lineal basado en la solución directa de un conjunto de ecuaciones lineales (después de una apropiada normalización de los datos).

2. Método iterativo que minimiza el error algebraico a la vez que satisface todas las restricciones del tensor.

3. Método iterativo que minimiza el error geométrico (el algoritmo "Gold Standard").

4. Método de cálculo exacto del tensor Trifocal dadas las proyecciones de 6 puntos en 3 cámaras.

5. Estimación robusta basada en RANSAC.

6.4.1. El tensor trifocal $\mathcal{T}$

El tensor trifocal juega un papel análogo en tres imágenes al que juega la matriz fundamental en dos. Dadas tres proyecciones, encapsula todas las relaciones geométricas (proyectivas) independientes de la estructura de la escena.

Las tres principales propiedades geométricas del tensor son: la homografía entre dos imágenes inducida por la reproyección de una recta (un plano) de la otra imagen; las relaciones entre correspondencias entre puntos e imágenes que provienen de relaciones de incidencia en el espacio; y la obtención de las matrices fundamentales y de proyección a partir del tensor.

Se puede utilizar el tensor para transferir puntos: si conocemos el tensor y la proyección de un punto en dos imágenes, podemos calcular la proyección correspondiente del punto en la tercera imagen. El tensor también se aplica a rectas: se puede calcular la posición de una recta en una imagen dadas las otras dos proyecciones y el tensor trifocal.

El tensor sólo depende del movimiento entre las cámaras y de los parámetros intrínsecos de las mismas y está definido unívocamente por las matrices de proyección. De todas formas, se puede calcular a partir de correspondencias de puntos, sin necesidad de conocer el movimiento o la calibración.

Grados de libertad

En su versión matricial, se puede ver el tensor trifocal como un conjunto de tres matrices de 3×3, así que consta de 27 elementos. Quitando el factor de proporcionalidad (común a las tres matrices), quedan 26 razones independientes. Sin embargo, el tensor tiene sólo 18 grados de libertad: una vez que se especifican 18 parámetros, los 27 elementos del tensor quedan determinados, salvo proporcionalidad. El número de grados de libertad se puede contar como sigue: cada una de las 3 matrices de proyección tiene 11 grados de libertad, lo que hace un total de 33; pero se deben restar 15 grados de libertad por la homografía arbitraria del espacio, quedando así 18 grados de libertad. El tensor satisface $26 - 18 = 8$ restricciones algebraicas independientes.

Relaciones trilineales

Un conjunto completo de ecuaciones trilineales sobre el tensor trifocal son las dadas en la tabla6.1.

Correspondencia	Relación	Número de ecuaciones
3 puntos	$x^i x'^j x''^k \epsilon_{jqs}\epsilon_{krt}\mathcal{T}_i^{qr} = 0_{st}$	4
2 puntos, 1 recta	$x^i x'^j l''_r \epsilon_{jqs}\mathcal{T}_i^{qr} = 0_s$	2
1 punto, 2 rectas	$x^i l'_q l''_r \mathcal{T}_i^{qr} = 0$	1
3 rectas	$l_p l'_q l''_r \epsilon^{piw}\mathcal{T}_i^{qr} = 0^w$	2

Cuadro 6.1: Relaciones trilineales entre coordenadas de puntos y rectas en las tres imágenes

La notación 0_{st} significa un tensor bidimensional con todos sus elementos nulos. El tensor de permutación ϵ_{ijk} es cero salvo que i, j y k sean distintos, y en este caso vale 1 o -1, dependiendo de si (ijk) es una permutación par o impar. Se utiliza el convenio de suma tensorial en la que si un superíndice (contravariante) se repite en subíndice (covariante) implica una suma en todos los valores de dicho índice: 1, 2 y 3.

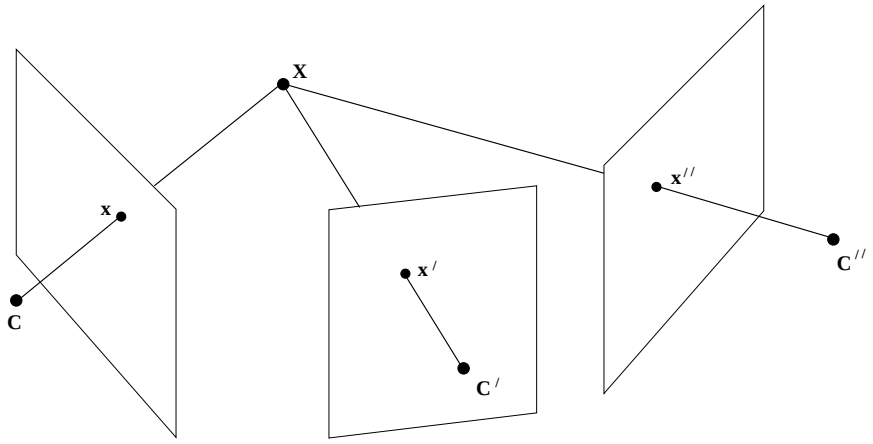


Figura 6.7: Relación de incidencia de 3 puntos correspondientes

6.4.2. Relación de $\mathcal{T}$ con las matrices de proyección

Es preferible explicar cómo obtener el tensor trifocal a partir de las matrices de proyección antes de aprender a estimarlo, porque se hará referencia en los algoritmos de estimación.

Dadas tres matrices de proyección $\mathtt{P} = \mathtt{A}$, $\mathtt{P}' = \mathtt{B}$ y $\mathtt{P}'' = \mathtt{C}$, existen tres tensores trifocales asociados, dependiendo de a qué proyección se de preferencia sobre las demás. El tensor trifocal $\mathcal{T}$ asociado a la primera matriz de proyección se obtiene según la fórmula de [1, pág. 404]. Notación: $\mathbf{a}^i$ es la i-ésima fila de la matriz $\mathtt{A}$, y del mismo modo, $\mathbf{b}^i$ es la i-ésima fila de $\mathtt{B}$, y $\mathbf{c}^i$ es la i-ésima fila de $\mathtt{C}$.

$$\mathcal{T}_i^{qr} = (-1)^{i+1} \det \begin{bmatrix} \sim \mathbf{a}^i \\ \mathbf{b}^q \\ \mathbf{c}^r \end{bmatrix} \tag{6.13}$$

donde $\sim \mathbf{a}^i$ significa la matriz $\mathtt{A}$ sin la fila i. En la fórmula, se omite la fila i de la matriz de proyección de la primera cámara, pero se incluyen las filas q y r de las otras dos matrices de proyección. Se da preferencia a la primera cámara porque contribuye con dos filas a cada determinante, en lugar de una.

6.4.3. Técnicas de estimación del tensor Trifocal

La última columna de la tabla 6.1 indica el número de ecuaciones linealmente independientes que genera el tipo de correspondencia de cada fila. Así, la primera línea de la tabla genera 9 ecuaciones, una para cada valor de s y t, pero sólo 4 de las 9 ecuaciones son linealmente independientes. Todas las ecuaciones son lineales en los elementos del tensor trifocal $\mathcal{T}$.

6.4.3.1. El algoritmo lineal

Dadas varias correspondencias de puntos o rectas entre las imágenes, el conjunto completo de ecuaciones es de la forma $\mathtt{A}\mathbf{t} = \mathbf{0}$, donde $\mathbf{t}$ es el vector de 27 elementos con los elementos del tensor. Las ecuaciones se pueden obtener de cualquier correspondencia de la tabla 6.1. Como $\mathcal{T}$ tiene 27 elementos, se necesitan 26 ecuaciones para obtener $\mathbf{t}$ salvo factor de proporcionalidad. Con más de 26 ecuaciones, se calcula una solución mínimo-cuadrática . Al igual que al calcular la matriz fundamental, se minimiza $\|\mathtt{A}\mathbf{t}\|$ sujeto a la restricción $\|\mathbf{t}\| = 1$ utilizando la SVD .

Habitualmente se utiliza sólo con puntos, por lo que hacen falta 7 o más correspondencias en 3 imágenes. Si se utilizan sólo correspondencias de rectas, hacen falta 14 o más de las mismas. Si se utilizan puntos y rectas a la vez, para que exista solución lineal se debe cumplir, según el número de ecuaciones indicado en la tabla 6.1:

$$4 \cdot \mathrm{N}^{\circ}\mathrm{puntos} + 2 \cdot \mathrm{N}^{\circ}\mathrm{rectas} \geq 27$$

Hemos indicado unas pinceladas de un algoritmo lineal para calcular el tensor trifocal . Sin embargo, para que el algoritmo sea práctico se deben normalizar los datos. Además, el tensor así estimado no tiene por qué satisfacer las restricciones algebraicas propias de un tensor trifocal.

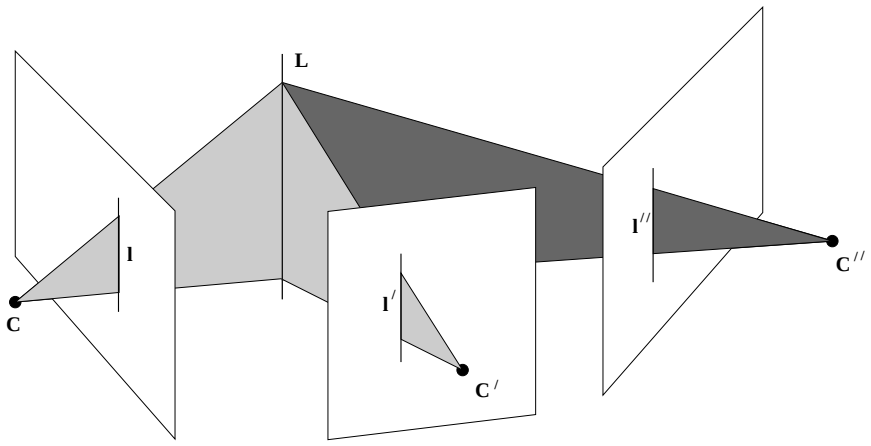


Figura 6.8: Relación de incidencia de 3 rectas correspondientes

Las restricciones internas del tensor $\mathcal{T}$

La diferencia más notable entre la matriz fundamental y el tensor trifocal es el mayor número de restricciones aplicables al último. La matriz fundamental sólo tiene una restricción, $\det \mathtt{F} = 0$, dejando 7 grados de libertad, descontando el factor de escala arbitrario. Por otro lado, el tensor trifocal tiene 27 elementos, pero sólo son necesarios 18 parámetros para especificar la configuración de cámaras equivalente, salvo homografía del espacio. Los elementos del tensor satisfacen 8 restricciones algebraicas independientes.

Definición. Se dice que un tensor trifocal $\mathcal{T}_i^{jk}$ es "geométricamente válido" o "satisface todas las restricciones internas" si existen 3 matrices de proyección $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}'$ y $\mathtt{P}''$ tales que $\mathcal{T}_i^{jk}$ se corre-

sponde con las tres matrices de proyección, de acuerdo con (6.13).

Al igual que en el caso de la matriz fundamental, es importante imponer estas restricciones de alguna forma para llegar a un tensor geométricamente válido. Si el tensor no satisface las restricciones y se utiliza para transferir un punto a la tercera imagen dada una correspondencia de puntos en las dos primeras, entonces la posición del punto transferido variará dependiendo de qué tipo de ecuación de la tabla anterior sea utilizada. Así pues, el objetivo es obtener un tensor geométricamente válido.

Las restricciones satisfechas por el tensor trifocal no se expresan tan fácilmente (como $\det = 0$), y algunos piensan que es un impedimento para calcular exactamente el tensor trifocal. Sin embargo, a la hora de calcular el tensor trifocal no es necesario expresar estas restricciones explícitamente. En vez de eso, son impuestas implícitamente por una parametrización apropiada del tensor trifocal, y no suelen causar problemas.

6.4.3.2. Minimización de la distancia algebraica

El algoritmo lineal, con o sin normalización de los datos, proporciona un tensor que puede no corresponderse con una configuración geométrica. La siguiente tarea es corregir el tensor para que satisfaga todas las restricciones internas necesarias: calcular un tensor trifocal válido $\mathcal{T}_i^{jk}$ a partir de correspondencias de puntos y/o correspondencias de rectas entre 3 imágenes. El tensor calculado minimizará el error algebraico asociado con los puntos observados.

Se desea minimizar $\|\mathtt{A}\hat{\mathbf{t}}\|$ sujeto a $\|\hat{\mathbf{t}}\| = 1$, donde $\hat{\mathbf{t}}$ es el vector con elementos de un tensor trifocal geométricamente válido. El algoritmo que consigue esto es el el algoritmo 15.2, [1, pág. 385] y es bastante similar al equivalente para estimar la matriz fundamental (§6.2.3.5). Al igual que entonces, el primer paso es calcular los epipolos, como en §6.2.3.5. Más referencias sobre este algoritmo son: §15.3 de [1] y [3].

Para que sea numéricamente estable, se encierra entre bloques de normalización de los datos y desnormalización del tensor trifocal, como en el algoritmo de los 8 puntos [5].

Funciones modelo y de coste
La función modelo de este problema es:

$$\begin{array}{rccl} f: & \mathbb{R}^6 & \longrightarrow & \mathbb{R}^{27} \\ & (\mathbf{e}_{21}, \mathbf{e}_{31}) & \mapsto & \epsilon = \mathtt{AEa} = \mathtt{A}\hat{\mathbf{t}} \end{array}$$

La matriz de restricción $\mathtt{E}$, de 27×18, se construye a partir de los epipolos. Esta matriz verifica $\mathrm{rank}(\mathtt{E}) = 15$ y sirve para imponer las restricciones del tensor trifocal. En $\mathbb{R}^{27}$, limitamos la búsqueda del tensor trifocal al subespacio generado por las columnas de $\mathtt{E}$.

$$\min_{\mathbf{a}} \|\mathtt{AEa}\| \text{ con la restriccion } \|\hat{\mathbf{t}}\| = \|\mathtt{Ea}\| = 1$$

Este problema se resuelve mediante el algoritmo A.3.7, § A3.4.4 de [1], que utiliza la SVD. La estimación inicial de los epipolos se obtiene mediante la estimación del tensor con el algoritmo lineal.

La función de coste (distancia algebraica) es la norma del vector de error o residuo ϵ, que indica cómo se verifican las ecuaciones obtenidas de las relaciones trilineales:

$$\text{coste} = \|\epsilon\| = \|\mathtt{AEa}\|.$$

Es aplicable el mismo comentario que en el caso de la matriz fundamental: “este algoritmo es bueno, pero tenemos otro mejor”.

6.4.3.3. Minimización del error de reproyección: Gold Standard

Al igual que en el cálculo de la matriz fundamental, esperamos obtener los mejores resultados con la solución de máxima verosimilitud (o "Gold Standard"). Así que discutamos el algoritmo 15.3, [1, pág. 386]: "The Gold Standard algorithm for estimating the Trifocal Tensor from image correspondences".

Dadas n correspondencias de puntos $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i \leftrightarrow \mathbf{x}''_i\}$ entre 3 imágenes, la función de coste a minimizar es

$$\sum_i \left[d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 + d(\mathbf{x}''_i, \hat{\mathbf{x}}''_i)^2\right] \tag{6.14}$$

donde los puntos $\hat{\mathbf{x}}_i$, $\hat{\mathbf{x}}'_i$, $\hat{\mathbf{x}}''_i$ satisfacen exactamente una restricción trifocal (como la de la tabla en 6.4.3) respecto del tensor trifocal estimado. Como en el caso de la matriz fundamental, se necesita introducir variables auxiliares de los puntos 3D y parametrizar el tensor trifocal por los elementos de las matrices de proyección $\mathtt{P}'$ y $\mathtt{P}''$.

Funciones modelo y de coste
La función modelo sirve para optimizar la función de coste, en el marco del algoritmo LM es:

$$f : \left[\frac{\mathbf{a}}{\mathbf{b}}\right] = \left[\begin{array}{c} \mathtt{P}' \\ \mathtt{P}'' \\ \hline \widehat{\mathbf{X}}_{a1} \\ \vdots \\ \widehat{\mathbf{X}}_{a\,n} \end{array}\right] \equiv \mathbf{P} \rightarrow \widehat{\mathbf{X}} \equiv \left[\begin{array}{c} \begin{pmatrix} \hat{\mathbf{x}}_{a1} \\ \hat{\mathbf{x}}'_{a1} \\ \hat{\mathbf{x}}''_{a1} \end{pmatrix} \\ \vdots \\ \begin{pmatrix} \hat{\mathbf{x}}_{a\,n} \\ \hat{\mathbf{x}}'_{a\,n} \\ \hat{\mathbf{x}}''_{a\,n} \end{pmatrix} \end{array}\right] = \left[\begin{array}{c} \widehat{\mathbf{X}}_{a1} \\ \vdots \\ \widehat{\mathbf{X}}_{a\,n} \end{array}\right]$$

Se minimiza la función de coste en el espacio de parámetros dado por las coordenadas afinesde los puntos 3D, $\hat{\mathbf{X}}_i$, y las dos matrices de proyección $\mathtt{P}'$ y $\mathtt{P}''$, con $\hat{\mathbf{x}}_i = [\mathtt{I} \mid \mathbf{0}]\hat{\mathbf{X}}_i$, $\hat{\mathbf{x}}'_i = \mathtt{P}'\hat{\mathbf{X}}_i$ y $\hat{\mathbf{x}}''_i = \mathtt{P}''\hat{\mathbf{X}}_i$. Esencialmente, se realiza un ajuste de haces sobre tres imágenes (§ 6.6.4). Casi se puede utilizar el ajuste de haces que más adelante explicaremos, si no fuera porque la primera matriz de proyección tiene que ser fija $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$.

El coste (distancia geométrica- error de reproyección) en términos de la función modelo se expresa:

$$\text{coste} = \|\mathbf{X} - f(\mathbf{P})\| = \sqrt{\sum_i \left[d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i, \hat{\mathbf{x}}'_i)^2 + d(\mathbf{x}''_i, \hat{\mathbf{x}}''_i)^2\right]}$$

La estimación inicial de un tensor trifocal válido, que proporciona el punto de partida de la búsqueda no lineal, se obtiene mediante el algoritmo de minimización de la distancia algebraica (§ 6.4.3.2).

Los mejores resultados se obtienen explotando el conocimiento del modelo, no optimizando la función de coste mediante técnicas generales. Al igual que en el caso de la matriz fundamental, durante la optimización se recomienda construir la matriz jacobiana de forma exacta a partir de los datos: (vector de parámetros $\mathbf{P}$ y vector de medidas estimado$f(\mathbf{P})$), siguiendo las fórmulas explicadas en § 6.6.4, salvo que se debe llevar cuidado con el convenio de mantener la primera matriz de proyección canónica $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$.

La matriz jacobiana de este problema es dispersa y tiene una estructura especial, ligeramente distinta a la del ajuste de haces que se verá en § 6.6.4. Su estructura está representada en la figura 6.9, junto con la matriz $\mathtt{J}^\top\mathtt{J}$, que hereda la estructura dispersa: las 2 submatrices $\mathtt{U}_j$ que componen la matriz $\mathtt{U}$ diagonal a bloques; las n submatrices $\mathtt{V}_i$ que forman la matriz $\mathtt{V}$ diagonal a bloques, y las submatrices $\mathtt{W}$ y $\mathtt{W}^\top$.

$$\mathtt{J} = \begin{pmatrix} \mathtt{A} & \mathtt{B} \end{pmatrix} \qquad\qquad \mathtt{J}^\top\mathtt{J} = \begin{pmatrix} \mathtt{U} & \mathtt{W} \\ \mathtt{W}^\top & \mathtt{V} \end{pmatrix}$$

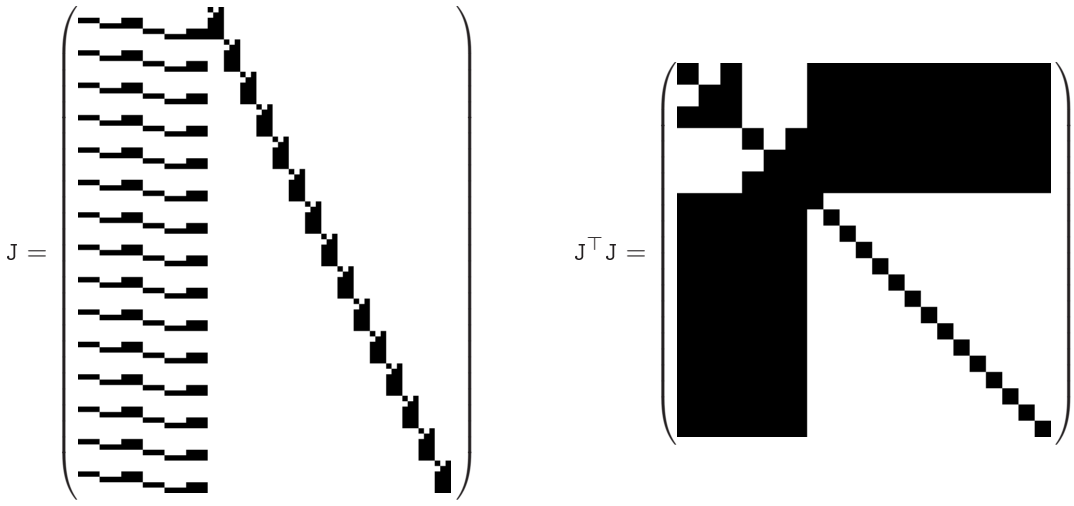


Figura 6.9: Estructura de las matrices jacobiana (J) y $\mathtt{J}^\top\mathtt{J}$ del algoritmo Gold Standard del Tensor Trifocal, con $n = 15$ puntos

En la submatriz A de la matriz jacobiana hay sólo dos franjas verticales, que son las variaciones de la función respecto de las matrices de proyección 2 y 3. Además, hay que dejar $2n$ filas nulas en esta submatriz debido a que sí existe una primera matriz de proyección, aunque sea fija. En la misma submatriz del problema del ajuste de haces proyectivo no hay ninguna fila nula.

6.4.3.4. Solución exacta con 6 correspondencias de puntos

Se puede calcular un tensor trifocal geométricamente válido a partir de 6 correspondencias de puntos en 3 imágenes, suponiendo que los puntos están en posición general. Hay una o tres posibles soluciones reales. El tensor se calcula dadas tres matrices de proyección, que se obtienen mediante el algoritmo 19.1, [1, pág. 493]. Esta solución mínima de 6 puntos se utiliza en el algoritmo robusto de § 6.4.3.5.

El algoritmo de los 6 puntos se basa en la dualidad Carlsson-Weinshall, que permite intercambiar el papel de los puntos visualizados por distintas cámaras con los propios centros ópticos de las cámaras. Este principio implica la posibilidad de dualizar los algoritmos de reconstrucción proyectiva para obtener nuevos algoritmos. Su explicación está fuera del alcance de esta memoria, para más detalles, se remite al lector a § 19.2.4 y § 19.2.5, de [1, pág. 492-493].

6.4.3.5. Estimación robusta mediante RANSAC

Aquí se describe un algoritmo para calcular la geometría trifocal entre tres imágenes, el cual incluye estimación robusta mediante RANSAC, siguiendo las recomendaciones de § 15.6, de [1, pág. 389]. No es exactamente el mismo que el algoritmo 15.4, ya que no realiza una clasificación previa de las correspondencias de puntos en *inliers* y *outliers* por parejas de imágenes mediante al algoritmo RANSAC para la matriz fundamental (§ 6.2.3.7).

El algoritmo se puede resumir en los siguientes pasos: dadas las correspondencias de puntos entre imágenes $(\mathbf{x}_i, \mathbf{x}'_i, \mathbf{x}''_i)$, $i = 1, \ldots, n$

1. **RANSAC básico**: Repetir hasta haber probado N muestras (donde N es determinado adaptativamente según el algoritmo 3.5, [1, pág. 105]) o haber logrado más de $T = (1 - \epsilon_{\text{fijo}})n$ puntos consistentes con el modelo (*inliers*) (ej. $\epsilon_{\text{fijo}} = 0{,}2$).

a) Seleccionar una muestra aleatoria de 6 parejas de puntos y calcular el tensor trifocal $\mathcal{T}$ mediante la dualidad Carlsson-Weinshall. (§ 6.4.3.4). Puede haber una o tres soluciones reales.

b) Calcular la distancia $d_\perp$ de cada punto al modelo de geometría trifocal indicado por $\mathcal{T}$.

c) Calcular el número de puntos consistentes con el modelo $\mathcal{T}$ como el número de correspondencias de puntos para los que $d_\perp < t = \sqrt{7{,}81}\sigma$ píxeles, siendo σ la desviación típica del ruido gaussiano esperado.

d) Si hay tres soluciones reales para $\mathcal{T}$, se calcula el número de *inliers* para cada solución, y se retiene la solución con mayor número de *inliers* (mayor soporte).

e) Comparar: Elegir el $\mathcal{T}$ con el mayor soporte. En el caso de empate, retener la solución con la menor desviación típica de *inliers* (en cuanto a distancia al modelo $d_\perp$).

f) Estimar la proporción ϵ de puntos no consistentes (*outliers*) y el número de muestras aleatorias a probar, N.

2. **Estimación óptima no lineal**: re-estimar $\mathcal{T}$ minimizando la distancia geométrica - Error de reproyección ec. (6.14) - a partir del mayor y mejor soporte encontrado. Se utiliza el algoritmo Gold Standard sólo sobre las correspondencias clasificadas como consistentes con el modelo.

3. **Identificación de nuevas parejas** consistentes con el modelo, igual que en los pasos c) y e). Continuar minimizando la distancia geométrica (paso 2) hasta que el soporte deje de crecer.

La distancia $d_\perp$ de cada punto al modelo es el error de reproyección. Para calcularla hace falta triangular los puntos mediante dos de las tres imágenes, obtener los puntos 3D y proyectarlos.

6.4.4. Calibración proyectiva de tres cámaras

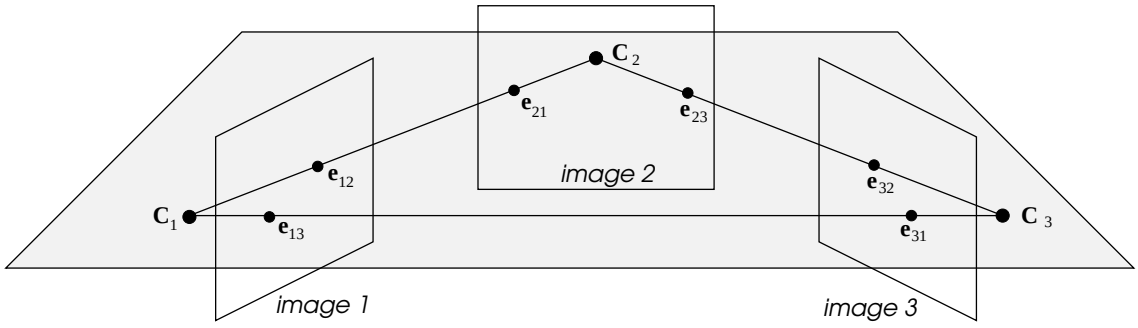


Figura 6.10: Geometría trifocal, plano trifocal y notación de los epipolos

Una vez conocido el tensor trifocal geométricamente válido $\mathcal{T}$, la aplicación directa consiste en encontrar tres matrices de proyección consistentes con el tensor. Este procedimiento se explica en el algoritmo 14.1, [1, pág. 366]. es importante prestar atención a la notación matricial en que aparece representado el tensor. El algoritmo tiene 3 pasos:

1. **Epipolos.** Llamamos $\mathbf{e}_{21} = \mathbf{e}'$ a la proyección del centro óptico de la primera cámara en la segunda imagen, y $\mathbf{e}_{31} = \mathbf{e}''$ a la proyección del centro óptico de la primera cámara en la tercera imagen. Es necesario calcular estos epipolos para seguir con el procedimiento.

2. Las **matrices fundamentales** de la cámara 1 hacia la 2 y hacia la 3 son:

$$\begin{aligned} \mathtt{F}_{21} &= [\mathbf{e}']_\times [\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3]\mathbf{e}'' &&= [\mathbf{e}_{21}]_\times [\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3]\mathbf{e}_{31} \\ \mathtt{F}_{31} &= [\mathbf{e}'']_\times [\mathtt{T}_1^\top, \mathtt{T}_2^\top, \mathtt{T}_3^\top]\mathbf{e}' &&= [\mathbf{e}_{31}]_\times [\mathtt{T}_1^\top, \mathtt{T}_2^\top, \mathtt{T}_3^\top]\mathbf{e}_{21} \end{aligned}$$

3. **Matrices de proyección**

$$\begin{array}{lcll} \mathtt{P} & = & [\mathtt{I} \mid \mathbf{0}] & \\ \mathtt{P}' & = & [[\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3]\mathbf{e}'' \mid \mathbf{e}'] & = & [[\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3]\mathbf{e}_{31} \mid \mathbf{e}_{21}] \\ \mathtt{P}'' & = & [(\mathbf{e}''\mathbf{e}''^{\top} - \mathtt{I})[\mathtt{T}_1^{\top}, \mathtt{T}_2^{\top}, \mathtt{T}_3^{\top}]\mathbf{e}' \mid \mathbf{e}''] & = & [(\mathbf{e}_{31}\mathbf{e}_{31}^{\top} - \mathtt{I})[\mathtt{T}_1^{\top}, \mathtt{T}_2^{\top}, \mathtt{T}_3^{\top}]\mathbf{e}_{21} \mid \mathbf{e}_{31}] \end{array}$$

La calibración proyectiva de las cámaras a partir del tensor trifocal no es necesaria para el algoritmo Gold Standard, ya que el propio vector de parámetros sobre el que se realiza la optimización contiene de forma transparente las matrices de proyección de las cámaras 2 y 3; la primera es la canónica.

6.5. Geometría de cuatro cámaras

Esta sección está dedicada a los algoritmos de § 16.3 de [1] y del artículo [2], sobre la estimación del tensor cuadrifocal Q por métodos numéricos dadas unas correspondencias de puntos entre cuatro imágenes o proyecciones. También se trata su relación con las matrices de proyección.

El desarrollo parecido a la sección sobre el tensor trifocal, utilizando técnicas similares. En concreto, se analizarán los siguientes algoritmos:

1. Método lineal basado en la solución directa de un conjunto de ecuaciones lineales en $\mathbb{R}^{81}$ (después de una apropiada normalización de los datos).
2. Método de Heyden (lineal, no iterativo).
3. Dos métodos iterativos que minimizan el error algebraico a la vez que satisfacen todas las restricciones del tensor.
   a) Método con una parametrización en $\mathbb{R}^9$.
   b) Método con una parametrización en $\mathbb{R}^{27}$.

6.5.1. El tensor cuadrifocal

El tensor cuadrifocal juega un papel análogo en cuatro imágenes al que juega la matriz fundamental en dos y el tensor trifocal en tres. El tensor cuadrifocal fue descubierto por Triggs y Heyden dio lugar a un algoritmo para utilizarlo en la reconstrucción.

Consideremos 4 cámaras $\mathtt{P}, \mathtt{P}', \mathtt{P}''$ y $\mathtt{P}'''$, y sea $\mathbf{X}$ un punto en el espacio que se proyecta en las 4 cámaras. Llamemos a los correspondientes puntos proyectados en las cuatro imágenes $\mathbf{u}, \mathbf{u}', \mathbf{u}''$ y $\mathbf{u}'''$, respectivamente. escribimos $\mathbf{x} \sim \mathtt{P}\mathbf{X}$ y relaciones parecidas para las otras proyecciones. Teniendo en cuenta los factores de escala, hay constantes k, k', k'' y k''' tales que $k\mathbf{u} = \mathtt{P}\mathbf{X}$, $k'\mathbf{u} = \mathtt{P}'\mathbf{X}$ y así para las otras dos imágenes. Este conjunto de ecuaciones se puede escribir en una sola ecuación matricial como sigue:

$$\begin{bmatrix} \mathtt{P} & \mathbf{u} & & & \\ \mathtt{P}' & & \mathbf{u}' & & \\ \mathtt{P}'' & & & \mathbf{u}'' & \\ \mathtt{P}''' & & & & \mathbf{u}''' \end{bmatrix} \begin{pmatrix} \mathbf{X} \\ -k \\ -k' \\ -k'' \\ -k''' \end{pmatrix} = \begin{bmatrix} \mathbf{a}^1 & u^1 & & & & & & \\ \mathbf{a}^2 & u^2 & & & & & & \\ \mathbf{a}^3 & u^3 & & & & & & \\ \mathbf{b}^1 & & u'^1 & & & & & \\ \mathbf{b}^2 & & u'^2 & & & & & \\ \mathbf{b}^3 & & u'^3 & & & & & \\ \mathbf{c}^1 & & & u''^1 & & & & \\ \mathbf{c}^2 & & & u''^2 & & & & \\ \mathbf{c}^3 & & & u''^3 & & & & \\ \mathbf{d}^1 & & & & u'''^1 & & & \\ \mathbf{d}^2 & & & & u'''^2 & & & \\ \mathbf{d}^3 & & & & u'''^3 & & & \end{bmatrix} \begin{pmatrix} \mathbf{X} \\ -k \\ -k' \\ -k'' \\ -k''' \end{pmatrix} = \mathbf{0} \qquad (6.15)$$

donde los vectores $\mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i$ y $\mathbf{d}^i$ son las filas de las matrices $\mathtt{P}, \mathtt{P}', \mathtt{P}''$ y $\mathtt{P}'''$, respectivamente.

Como esta ecuación tiene una solución, la matriz $\mathcal{X}$ de la izquierda tiene como mucho rango 7, y por lo tanto todos los determinantes de 8×8 son nulos. Cualquier determinante que contiene menos de dos filas de cada matriz de proyección da lugar a una relación trilineal o bilineal en las restantes matrices de proyección. Sucede algo diferente cuando consideramos determinantes de 8×8 que contienen 2 filas de cada matriz de proyección. Tal determinante da lugar a una relación cuadrilineal de la forma

$$u^i u'^j u''^k u'''^l \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry}\epsilon_{lsz} Q^{pqrs} = 0_{wxyz} \tag{6.16}$$

donde cada elección de las variables w, x, y, z da una ecuación distinta y 0_{wxyz} es un tensor nulo con cuatro índices. El tensor de cuatro dimensiones Q^{pqrs} llamado *tensor cuadrifocal* se define

$$Q^{pqrs} = \det \begin{bmatrix} \mathbf{a}^p \\ \mathbf{b}^q \\ \mathbf{c}^r \\ \mathbf{d}^s \end{bmatrix} \tag{6.17}$$

Obsérvese que los cuatro índices del tensor cuadrifocal son contravariantes; no hay proyección privilegiada como en el caso del tensor trifocal. Sólo hay un tensor cuadrifocal correspondiente a las cuatro imágenes.

6.5.2. Técnicas de estimación del tensor cuadrifocal

Cada correspondencia de puntos entre las cuatro imágenes da lugar vía la ec. (6.16) a 81 ecuaciones lineales en los elementos de Q, una ecuación para cada elección de los índices w, x, y, z. De las 81 ecuaciones sólo 16 son linealmente independientes. Dadas suficientes correspondencias entre las 4 imágenes se obtiene un conjunto de ecuaciones lineales de la forma

$$\mathtt{A}\mathbf{q} = \mathbf{0} \tag{6.18}$$

donde $\mathbf{q}$ es el vector que contiene los 81 elementos de Q. Como Q está definido salvo factor de proporcionalidad, parece que con 5 correspondencias de puntos hay suficientes ecuaciones para calcular Q: $(16\times 5 = 80)$. Sin embargo, conjuntos de ecuaciones de distintas correspondencias de puntos verifican ciertas relaciones lineales. Por lo que hacen falta 6 correspondencias de puntos para hallar Q. Con 6 o más puntos se debe resolver un sistema lineal por mínimos cuadrados para hallar Q.

También es posible deducir ecuaciones para hallar el tensor cuadrifocal a partir de correspondencias de rectas o correspondencias mixtas de puntos y rectas. Las relaciones están recogidas en la tabla 6.2.

Correspondencia	Relación	Número de ecuaciones
4 puntos	$u^i u'^j u''^k u'''^l \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry}\epsilon_{lsz} Q^{pqrs} = 0_{wxyz}$	16
3 puntos, 1 recta	$u^i u'^j u''^k l'''_s \epsilon_{ipw}\epsilon_{jqx}\epsilon_{kry} Q^{pqrs} = 0_{wxy}$	8
2 puntos, 2 rectas	$u^i u'^j l''_r l'''_s \epsilon_{ipw}\epsilon_{jqx} Q^{pqrs} = 0_{wx}$	4
3 rectas	$l_p l'_q l''_r Q^{pqrs} = 0^s$	3
4 rectas	$l_p l'_q l''_r Q^{pqrs} = 0^s$, $l_p l'_q l'''_s Q^{pqrs} = 0^r$, ...	9

Cuadro 6.2: Relaciones cuadrilineales entre coordenadas de puntos y rectas en las cuatro imágenes

6.5.2.1. El algoritmo lineal

En presencia de ruido, no suele existir una solución exacta de un sistema sobredeterminado del tipo (6.18) como los obtenidos con correspondencias de puntos entre las imágenes. En su lugar, podemos emplear el algoritmo de mínimos cuadrados para hallar el vector $\mathbf{q}$ de norma unidad que minimiza la distancia algebraica $\|\mathtt{A}\mathbf{q}\|$. El vector $\epsilon = \mathtt{A}\mathbf{q}$ es el vector de error y su norma es lo que se desea minimizar. La solución es la columna de $\mathtt{V}$ (vector singular unitario) correspondiente al menor valor singular de $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^\top$, es decir la última columna (número 81).

Consideremos una estimación del tensor cuadrifocal Q, representada por un vector $\mathbf{q}$, y sea $\mathtt{A}$ la matriz de las ecuaciones de un conjunto de correspondencias de puntos $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i \leftrightarrow \mathbf{u}''_i \leftrightarrow \mathbf{u}'''_i$ entre 4 imágenes. El vector $\mathtt{A}\mathbf{q}$ es el vector de error algebraico asociado a la estimación $\mathbf{q}$, relativo a las medidas en $\mathtt{A}$. Nuestro objetivo es encontrar el vector unitario $\mathbf{q}$ que minimiza $\|\mathtt{A}\mathbf{q}\|$. Podemos buscar el vector $\mathbf{q}$ en todo el espacio $\mathbb{R}^{81}$ o restringir la búsqueda a un subconjunto del ambiente, en caso de imponer restricciones.

Restricciones

La forma más sencilla de estimar el tensor Q es el método lineal de los mínimos cuadrados, que encuentra el vector unitario $\mathbf{q}$ que minimiza la distancia la algebraica, permitiendo que $\mathbf{q}$ varíe en todo el espacio ambiente R^{81}. Al igual que en el caso de la matriz fundamental y el tensor trifocal, el tensor Q que uno encuentra de esta forma puede no corresponderse con una configuración de 4 matrices de proyección. Esto sólo se cumplirá si Q cumple ciertas restricciones. Un tensor cuadrifocal Q está determinado por cuatro matrices de proyección, como indica la fórmula (6.17). Las cuatro matrices tienen un total de 44 grados de libertad. Sin embargo, el tensor cuadrifocal (al igual que la matriz fundamental y el tensor trifocal) es invariante a una transformación proyectiva del espacio, ya que su valor está únicamente determinado por los puntos proyectados en las imágenes. El tensor cuadrifocal sólo depende de 29 parámetros esenciales. Por lo tanto, debe satisfacer $51 = 80 - 29$ restricciones, además de la ambigüedad del factor de escala. Como se puede apreciar, los 81 elementos del tensor cuadrifocal son muy redundantes, en lo que se refiere a la descripción de la configuración de cámaras.

Por contra, la matriz fundamental tiene elementos salvo escalado y debe satisfacer una sola restricción: $\det \mathtt{F} = 0$. El tensor trifocal tiene 27 elementos y debe satisfacer 8 restricciones, ya que la configuración proyectiva de las cámaras tiene $18 = 33 - 15$ grados de libertad. Aunque uno pueda esperar obtener resultados razonables ignorando las restricciones en la matriz fundamental e incluso en el tensor trifocal, está claro que no se pueden ignorar las 51 restricciones del tensor cuadrifocal y esperar obtener resultados razonables.

El método descrito en [2] para el cálculo del tensor cuadrifocal es encontrar el vector $\mathbf{q}$ que minimiza la distancia algebraica a la vez que proviene de una configuración de 4 cámaras. Es decir, buscar el mínimo dentro de la variedad de los tensores cuadrifocales *geométricamente válidos*, contenida en $\mathbb{R}^{81}$. Para ello utiliza el algoritmo A.3.7, § A3.4.4 de [1] de minimización con restricciones lineales.

La matriz de medida reducida

En general, la matriz $\mathtt{A}$ tendrá un gran número de filas. Es posible reemplazar $\mathtt{A}$ por una matriz cuadrada $\hat{\mathtt{A}}$ tal que $\|\mathtt{A}\mathbf{q}\| = \|\hat{\mathtt{A}}\mathbf{q}\|$ para cualquier vector $\mathbf{q}$. Tal matriz recibe el nombre de matriz de medida reducida. Una forma eficiente de calcular $\hat{\mathtt{A}}$ es utilizar la descomposición QR: $\mathtt{A} = \mathtt{Q}\hat{\mathtt{A}}$, donde $\mathtt{Q}$ tiene columnas ortogonales y $\hat{\mathtt{A}}$ es triangular superior y cuadrada.

De esta forma, toda la información que se necesita guardar de un conjunto de correspondencias $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i \leftrightarrow \mathbf{u}''_i \leftrightarrow \mathbf{u}'''_i$ está contenida en la matriz $\hat{\mathtt{A}}$. Minimizar la distancia algebraica $\|\mathtt{A}\mathbf{q}\|$ es equivalente a minimizar $\|\hat{\mathtt{A}}\mathbf{q}\|$, sin embargo la última requiere menos memoria y menos operaciones.

6.5.2.2. El algoritmo de Heyden

Un método presentado por Heyden para reducir el número de restricciones incumplidas sobre el tensor cuadrifocal utiliza el tensor cuadrifocal reducido. Heyden también aplicó esta técnica para definir una matriz fundamental reducida y un tensor trifocal reducido, como se describe en [21], [22] y [24].

El tensor cuadrifocal reducido

Supongamos que entre el conjunto de correspondencias, seleccionamos 3 de ellas: $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i \leftrightarrow \mathbf{u}''_i \leftrightarrow \mathbf{u}'''_i$ para $i = 1 \ldots 3$. Esta selección debería hacerse de tal forma que los puntos $\mathbf{u}_1, \mathbf{u}_2$ y $\mathbf{u}_3$ no están alineados, ni tampoco sus correspondientes puntos en las otras imágenes. Existe una transformación proyectiva $\mathtt{T}$ que transforma los puntos de coordenadas homogéneas $\mathbf{e}_1 = (1,0,0)^\top$, $\mathbf{e}_2 = (0,1,0)^\top$ y $\mathbf{e}_3 = (0,0,1)^\top$ en los puntos $\mathbf{u}_1, \mathbf{u}_2$ y $\mathbf{u}_3$. Obsérvese que $\mathtt{T}$ no es una transformación afín, ya que no deja invariante la recta del infinito. Por lo tanto, no queda definida por las imágenes de 3 puntos, sin embargo una elección sencilla de la matriz $\mathtt{T}$ es

$$\mathtt{T} = [\mathbf{u}_1\, \mathbf{u}_2\, \mathbf{u}_3] \tag{6.19}$$

Es fácil verificar que $\mathtt{T}\mathbf{e}_i = \mathbf{u}_i$. Supongamos que se aplica la transformación $\mathtt{T}^{-1}$ a cada punto $\mathbf{u}_i$ de la imagen, dando lugar a un nuevo conjunto de puntos y llamemos a estos puntos transformados con el mismo símbolo $\mathbf{u}_i$. Así pues, se dispone de un conjunto de puntos $\mathbf{u}_i$ de los cuales los tres primeros $\mathbf{u}_1, \mathbf{u}_2$ y $\mathbf{u}_3$ son iguales a $\mathbf{e}_i$, $i = 1, \ldots, 3$.

Llamemos $\mathbf{X}_i$ a los puntos del espacio correspondientes a los puntos proyectados $\mathbf{u}_i$. Se cumple $\mathbf{u}_i = \mathtt{P}\mathbf{X}_1$. Centrémonos en los tres primeros puntos $\mathbf{X}_1, \mathbf{X}_2$ y $\mathbf{X}_3$. Como las proyecciones $\mathbf{u}_1, \mathbf{u}_2$ y $\mathbf{u}_3$ no están alineados (por hipótesis), tampoco lo están los puntos $\mathbf{X}_i$ que se proyectan en ellos. Por lo tanto, es posible seleccionar una referencia proyectiva en la que estos puntos tengan coordenadas $\mathbf{X}_1 = (1,0,0,0)^\top$, $\mathbf{X}_2 = (0,1,0,0)^\top$ y $\mathbf{X}_3 = (0,0,1,0)^\top$. Como estos puntos se proyectan sobre los puntos $\mathbf{e}_i$, se verifica que la forma de la matriz de proyección es

$$\mathtt{P} = \begin{bmatrix} p_1 & & & q_1 \\ & p_2 & & q_2 \\ & & p_3 & q_3 \end{bmatrix} \tag{6.20}$$

Supongamos, además que el centro óptico de la cámara está en el punto $(0,0,0,1)^\top$, lo que significa que $\mathtt{P}(0,0,0,1)^\top = (0,0,0)^\top$. Por lo tanto, $q_1 = q_2 = q_3 = 0$.

Aplicando un argumento similar a cada una de las otras cámaras, se observa que $\mathtt{P}, \mathtt{P}', \mathtt{P}''$ y $\mathtt{P}'''$ tienen una forma similar, que consiste en un bloque diagonal a la izquierda más una cuarta columna arbitraria. Finalmente, uno puede multiplicar cada una de las matrices de proyección por la izquierda por la matriz

$$\begin{bmatrix} p_1^{-1} & & & -p_1^{-1}q_1 \\ & p_2^{-1} & & -p_2^{-1}q_2 \\ & & p_3^{-1} & -p_3^{-1}q_3 \\ & & & 1 \end{bmatrix}$$

Esto hace que la primera matriz $\mathtt{P}$ sea la canónica $[\mathtt{I} \mid \mathbf{0}]$, mientras se conserva la estructura (6.20) de las otras matrices de proyección.

Notación: Para evitar tener que contar el número de primas para distinguir los elementos de las tres

cámaras, las escribiremos de la siguiente forma:

$$\mathtt{P} = [\mathtt{I} \mid \mathbf{0}] \quad \mathtt{P}' = \begin{bmatrix} a_1 & & & a'_1 \\ & a_2 & & a'_2 \\ & & a_3 & a'_3 \end{bmatrix}$$
$$\mathtt{P}'' = \begin{bmatrix} b_1 & & & b'_1 \\ & b_2 & & b'_2 \\ & & b_3 & b'_3 \end{bmatrix} \quad \mathtt{P}''' = \begin{bmatrix} c_1 & & & c'_1 \\ & c_2 & & c'_2 \\ & & c_3 & c'_3 \end{bmatrix} \tag{6.21}$$

Ahora consideremos el tensor cuadrifocal Q^{ijkl} definido por (6.17) en término de las matrices (6.21). Este tensor se conoce como *tensor cuadrifocal reducido*. Cada elemento se define mediante el determinante construido con una fila de cada una de las 4 matrices de proyección. Obsérvese que si uno de los índices 1, 2 o 3 no está presente en el conjunto $\{i, j, k, l\}$ de Q^{ijkl}, entonces, la columna del índice que falta es cero y el elemento Q^{ijkl} es cero. Así, los únicos elementos no nulos son aquellos en los que se hallan los tres números 1, 2 y 3 en los índices. Como hay cuatro índices, uno debe estar repetido. Con estas consideraciones, sólo hay 36 elementos no nulos en el tensor cuadrifocal reducido: 6 para contar qué pareja de índices son iguales y 6 para contar el número de permutaciones de los índices 1,2,3.

Resumamos el algoritmo debido a Heyden para calcular el tensor cuadrifocal reducido a partir de un conjunto de correspondencias de puntos.

1. Seleccionar tres correspondencias de puntos $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i \leftrightarrow \mathbf{u}''_i \leftrightarrow \mathbf{u}'''_i$ para $i = 1, \ldots, 3$ y calcular las transformaciones $\mathtt{T}, \mathtt{T}', \mathtt{T}''$ y $\mathtt{T}'''$ que transforman los puntos $\mathbf{e}_i$ $i = 1, \ldots, 3$ en los puntos seleccionados. Aplicar las transformaciones inversas a todos los puntos en cada imagen para obtener un nuevo conjunto de correspondencias transformadas.

2. Cada correspondencia transformada entre puntos proporciona un conjunto de ecuaciones lineales en los 36 elementos no nulos del tensor cuadrifocal reducido, de acuerdo a (6.16). Se ignoran las tres primeras correspondencias de puntos, pues darán ecuaciones nulas. Resolver este sistema de ecuaciones por mínimos cuadrados para encontrar el tensor cuadrifocal reducido.

3. Pasar del tensor cuadrifocal reducido al tensor cuadrifocal. El tensor cuadrifocal correspondiente al conjunto inicial de puntos (no los transformados mediante $\mathtt{T}$, etc.) se pude obtener transformando el tensor cuadrifocal reducido estimado en el paso anterior. Hay que considerar cómo se transforma el tensor cuadrifocal. El tensor cuadrifocal tiene 4 índices contravariantes (superíndices). Esto quiere decir que Q se transforma del mismo modo que los puntos. En particular, denotemos por $\hat{\mathbf{u}}_i$ a los puntos transformados, $\hat{\mathbf{u}}_i = \mathtt{T}^{-1}\mathbf{u}_i$ y definimos $\hat{\mathbf{u}}'_i, \hat{\mathbf{u}}''_i$ y $\hat{\mathbf{u}}'''_i$ de forma análoga. Supongamos que $\hat{Q}$ es el tensor cuadrifocal estimado según las correspondencias de puntos $\hat{\mathbf{u}}_i \leftrightarrow \hat{\mathbf{u}}'_i \leftrightarrow \hat{\mathbf{u}}''_i \leftrightarrow \hat{\mathbf{u}}'''_i$, y Q es el de las correspondencias $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i \leftrightarrow \mathbf{u}''_i \leftrightarrow \mathbf{u}'''_i$. Como $\mathbf{u}_i = \mathtt{T}\hat{\mathbf{u}}_i$, el tensor $\hat{Q}$ también se transforma mediante $\mathtt{T}$. En concreto, se verifica

$$Q^{ijkl} = \hat{Q}^{abcd}\mathtt{T}^i_a\mathtt{T}'^j_b\mathtt{T}''^k_c\mathtt{T}'''^l_d \tag{6.22}$$

   Si $\hat{Q}$ es el tensor estimado a partir de los puntos transformados, entonces (6.22) permite obtener el tensor correspondiente a los puntos originales.

Extracción de las matrices de proyección

Siguiendo a Heyden, damos un método para calcular las matrices de proyección una vez que el tensor cuadrifocal reducido ha sido calculado. Al contrario que los métodos para calcular las matrices de proyección a partir de la matriz fundamental o el tensor trifocal, en los que primero se calculan los epipolos, ahora se calculan los elementos de las diagonales de (6.21), y después se calculan las columnas finales, que representan los epipolos. Para entender esto, consideremos dos elementos Q^{2311} y Q^{3211}.

De (6.17) se deduce que

$$Q^{2311} = \det \begin{bmatrix} & 1 & & \\ & & a_3 & a'_3 \\ b_1 & & & b'_1 \\ c_1 & & & c'_1 \end{bmatrix} = a_3(b_1c'_1 - c_1b'_1)$$

Del mismo modo, se obtiene que $Q^{3211} = -a_2(b_1c'_1 - c_1b'_1)$. Entonces, la razón $a_3 : a_2 = Q^{2311} : -Q^{3211}$. Siguiendo de esta manera, una forma de obtener los elementos de las matrices (6.21) es resolver las siguientes ecuaciones:

$$\begin{bmatrix} 0 & Q^{2311} & Q^{3211} \\ Q^{1322} & 0 & Q^{3122} \\ Q^{1233} & Q^{2133} & 0 \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \mathbf{0}$$

$$\begin{bmatrix} 0 & Q^{2131} & Q^{3121} \\ Q^{1232} & 0 & Q^{3212} \\ Q^{1323} & Q^{2313} & 0 \end{bmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \mathbf{0} \tag{6.23}$$

$$\begin{bmatrix} 0 & Q^{2113} & Q^{3112} \\ Q^{1223} & 0 & Q^{3221} \\ Q^{1332} & Q^{2331} & 0 \end{bmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \mathbf{0}$$

Estas ecuaciones tienen un patrón: el primer índice de Q^{ijkl} se corresponde con la columna de cada matriz en (6.23), y el índice repetido a la fila. De esta forma se calcula los elementos diagonales de (6.21). Se debe elegir cada vector solución $(a_1, a_2, a_3)^\top$, $(b_1, b_2, b_3)^\top$ y $(c_1, c_2, c_3)^\top$ de norma unidad.

Extracción de los epipolos

Una vez que se conocen los valores de $(a_1, a_2, a_3)^\top$, $(b_1, b_2, b_3)^\top$ y $(c_1, c_2, c_3)^\top$, los elementos de Q son lineales en las entradas de las matrices de proyección reducidas $a'_1, a'_2, a'_3, b'_1, b'_2, b'_3, c'_1, c'_2, c'_3$. Esto es así porque estas entradas aparecen en la última columna del determinante (6.17) que representa el elemento Q^{ijkl}. Como sólo aparecen en una columna, el determinante no puede dar elementos de mayor grado que 1 en esas entradas. Expresaremos esta relación lineal de la forma $\hat{\mathbf{q}} = \mathtt{M}\mathbf{a}'$, donde $\hat{\mathbf{q}}$ es el vector con los elementos del tensor cuadrifocal reducido y $\mathbf{a}'$ es el vector $\mathbf{a}' = (a'_1, a'_2, a'_3, b'_1, b'_2, b'_3, c'_1, c'_2, c'_3)^\top$. La solución de mínimos cuadrados de este sistema da los elementos de $\mathbf{a}'$, que junto con los elementos de $\mathbf{a} = (a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3)^\top$ antes calculados determinan las matrices de proyección reducidas, de acuerdo con (6.21).

El algoritmo completo (debido a Heyden) para calcular las matrices de proyección a partir del tensor cuadrifocal reducido es el siguiente:

1. Calcular $\mathbf{a}$: los elementos de las diagonales de las matrices de proyección reducidas resolviendo las ecuaciones (6.23).

2. Calcular $\mathbf{a}'$. Expresar los elementos del tensor cuadrifocal reducido en función del vector $\mathbf{a}'$, con las últimas columnas de las matrices de proyección reducidas. Esto da un sistema de ecuaciones $\hat{\mathbf{q}} = \mathtt{M}\mathbf{a}'$, donde los elementos de $\mathtt{M}$ son términos cuadráticos en los elementos de $\mathbf{a}$. Resolver este sistema de ecuaciones por mínimos cuadrados, para conocer $\mathbf{a}'$.

3. Si se precisa, el tensor cuadrifocal reducido se puede calcular según (6.17). Finalmente, el tensor cuadrifocal correspondiente a los datos originales se calcula según (6.22). De forma alternativa, si sólo se necesitan las matrices de proyección originales, entonces se calculan transformando las matrices de proyección reducidas estimadas en los pasos 1 y 2.

$$\mathbf{u}_i = \mathtt{T}\hat{\mathbf{u}}_i \Longrightarrow \mathtt{P} = \mathtt{T}\hat{\mathtt{P}}$$

Es importante notar que este método, junto con el algoritmo para para encontrar el tensor cuadrifocal reducido proporcionan un método de calcular un tensor cuadrifocal geométricamente válido: que se corresponde con una correcta elección de las cámaras, y por lo tanto satisface todas las restricciones necesarias. Esto se consigue, por supuesto, sin describir explícitamente la forma de las restricciones.

6.5.2.3. Minimización de la distancia algebraica

El algoritmo dad en la última sección presenta varias desventajas desde un punto de vista computacional.

1. Al calcular las transformaciones $\mathtt{T}, \ldots, \mathtt{T}'''$ de acuerdo con (6.19) uno se arriesga a encontrar un caso en el que los tres puntos utilizados para definir la transformación estén casi alineados. En esta caso, $\mathtt{T}$ está cerca de ser singular, y por lo tanto no invertible. Esto significa que las posiciones de los puntos transformados calculados al aplicar $\mathtt{T}^{-1}$ a los datos originales puede que no sean muy estables.

2. Al calcular los epipolos, las matrices de (6.23) puede que no sean singulares. En la práctica, la solución se halla calculando la matriz singular más cercana (en el sentido de la norma inducida) y tomando el núcleo de dicha matriz. Sin embargo, no hay una justificación teórica de para adoptar este criterio de cercanía. El problema es análogo al encontrado al imponer la restricción de singularidad para la matriz fundamental.

3. En el cálculo del vector $\mathbf{a}'$, una vez más se resuelve un sistema por mínimos cuadrados. Sin embargo, una vez más, la cantidad que se está minimizando no tiene una interpretación significativa en términos del error de los datos originales.

A continuación se muestra cómo evitar estos problemas de forma que se minimice la distancia algebraica. El resultado es que en los tres pasos del siguiente algoritmo se está minimizando la misma función de coste algebraico, y mejora el rendimiento numéricamente.

Algoritmo

La fórmula de transformación (6.22) del tensor es lineal en los elementos de $\hat{Q}$. Más específicamente, podemos escribir $\mathbf{q} = \mathtt{E}\hat{\mathbf{q}}$. En esta ecuación, $\hat{\mathbf{q}}$ y $\mathbf{q}$ son vectores con los elementos de los tensores cuadrifocales reducido y completo, respectivamente, y $\mathtt{E}$ es una matriz de 81×36. Sea $\hat{\mathtt{A}}$ la matriz de medida reducida calculada a partir de las observaciones (puntos) *originales*. Es deseable calcular el vector unitario $\mathbf{q}$ que minimiza el error algebraico $\|\hat{\mathtt{A}}\mathbf{q}\|$ sujeto a la restricción de que $\mathbf{q}$ proviene de un tensor cuadrifocal reducido, según la restricción $\mathbf{q} = \mathtt{E}\hat{\mathbf{q}}$. En otras palabras queremos resolver

$$\min_{\hat{\mathbf{q}}} \|\hat{\mathtt{A}}\mathtt{E}\hat{\mathbf{q}}\| \quad \text{sujeto a} \quad \|\mathtt{E}\hat{\mathbf{q}}\| = 1$$

Este es un problema del tipo que resuelve el algoritmo A.3.7, § A3.4.4 de [1]. Es importante observar que la transformación $\hat{Q} \mapsto Q$ dada por (6.22) depende de las matrices $\mathtt{T}, \ldots, \mathtt{T}'''$ dadas por fórmulas como (6.19), construidas a partir de las coordenadas de las correspondencias de puntos. En todo el proceso, nunca es necesario invertir estas transformaciones. Además, la matriz de medida reducida $\hat{\mathtt{A}}$ se forma a partir de los puntos originales no transformados. Así que se calcula el tensor cuadrifocal reducido a la vez que se evita por completo el problema de invertir las matrices, que podía ser potencialmente cuasi-singular.

No parece posible encontrar los elementos de las diagonales de las matrices de proyección reducidas de forma que se minimice el error algebraico y de forma lineal. Por consiguiente, se utiliza el método de Heyden para encontrar la solución mínimo-cuadrática de las ecuaciones (6.23). Sólo resta el problema de encontrar las últimas columnas de las matrices, es decir, el vector $\mathbf{a}'$. Como en aquel método, se puede expresar el tensor cuadrifocal reducido $\hat{Q}$ linealmente en función de $\mathbf{a}'$, escribiendo $\hat{\mathbf{q}} = \mathtt{M}\mathbf{a}'$, suponiendo que conocemos las diagonales de las matrices (vector $\mathbf{a}$) y con ellas construimos $\mathtt{M}$.

Ahora queremos encontrar el vector unitario $\mathbf{q}$ que minimiza $\|\hat{\mathtt{A}}\mathbf{q}\|$ sujeto a las restricciones $\mathbf{q} = \mathtt{E}\hat{\mathbf{q}}$ y $\hat{\mathbf{q}} = \mathtt{M}\mathbf{a}'$. Estas dos restricciones se pueden poner como una sola $\mathbf{q} = \mathtt{EM}\mathbf{a}'$, así que el problema se puede enunciar como sigue:

$$\min_{\mathbf{a}'} \|\hat{\mathtt{A}}(\mathtt{EM})\mathbf{a}'\| \quad \text{sujeto a} \quad \|(\mathtt{EM})\mathbf{a}'\| = 1$$

Una vez más, este es un problema de los que resuelve el algoritmo A.3.7, § A3.4.4 de [1]. La solución proporciona los valores de $\mathbf{a}'$ o $\mathbf{q}$. Así que podemos calcular el tensor cuadrifocal directamente calculando $\mathbf{q}$, o podemos extraer las matrices de proyección reducidas mediante $\mathbf{a}'$.

El algoritmo completo propuesto en [2] es:

1. Normalización afín de los datos: puntos observados.

$$\mathbf{u} = \mathtt{H}\mathbf{x}, \mathbf{u}' = \mathtt{H}'\mathbf{x}', \ldots$$

2. Estimación del tensor cuadrifocal normalizado

   a) Construir la matriz de medida reducida $\hat{\mathtt{A}}$ a partir de los datos $\mathbf{u}, \mathbf{u}', \ldots$

   b) Obtener las matrices de las transformaciones $\mathtt{T}, \ldots, \mathtt{T}'''$ a partir de tres correspondencias mediante la fórmula (6.19).

   c) Calcular la matriz $\mathtt{E}$ de 81×36 de la transformación, tal que $\mathbf{q} = \mathtt{E}\hat{\mathbf{q}}$ según la regla de transformación (6.22).

   d) Resolver el problema de minimización:

$$\min_{\hat{\mathbf{q}}} \|\hat{\mathtt{A}}\mathtt{E}\hat{\mathbf{q}}\| \quad \text{sujeto a} \quad \|\mathtt{E}\hat{\mathbf{q}}\| = 1$$

   para encontrar $\hat{\mathbf{q}}$, una estimación inicial del tensor cuadrifocal reducido.

   e) Calcular los elementos diagonales de las matrices de proyección reducidas (vector $\mathbf{a}$) resolviendo (6.23).

   f) Calcular la matriz $\mathtt{M}$ de 36×9 tal que $\hat{\mathbf{q}} = \mathtt{M}\mathbf{a}'$, donde $\mathbf{a}'$ es el vector de $\mathbb{R}^9$ con los elementos de las últimas columnas de las matrices de proyección reducidas. Estas columnas representan los epipolos relativos a la primera cámara.

   g) Resolver el problema de minimización:

$$\min_{\mathbf{a}'} \|\hat{\mathtt{A}}(\mathtt{EM})\mathbf{a}'\| \quad \text{sujeto a} \quad \|(\mathtt{EM})\mathbf{a}'\| = 1$$

   De aquí se obtiene el vector $\mathbf{q} = \mathtt{EM}\mathbf{a}'$, que es el tensor cuadrifocal completo, normalizado.

   h) Calcular las matrices de proyección reducidas (6.21) a partir de los vectores $\mathbf{a}$ y $\mathbf{a}'$. Transformar las matrices de proyección multiplicando por la izquierda por las matrices $\mathtt{T}, \ldots, \mathtt{T}'''$.

$$\mathtt{P} = \mathtt{T}\hat{\mathtt{P}}, \mathtt{P}' = \mathtt{T}'\hat{\mathtt{P}}', \ldots$$

3. Desnormalización afín ...

   a) del tensor cuadrifocal según las matrices $\mathtt{H}, \ldots, \mathtt{H}'''$.

   b) de las matrices de proyección, multiplicando por la izquierda por las matrices $\mathtt{H}$.

$$\mathbf{x}_i = \mathtt{H}^{-1}\mathbf{u}_i \Longrightarrow \mathtt{P}_{\mathbf{x}} = \mathtt{H}^{-1}\mathtt{P}$$

El tensor cuadrifocal hallado de esta forma es un tensor válido que satisface todas restricciones propias, ya que ha sido calculado a partir de una parametrización de las matrices de proyección. Las propias matrices de proyección están tan unidas al tensor cuadrifocal reducido que a la vez que se desnormaliza el tensor, también se pueden desnormalizar las matrices de proyección, por lo que la calibración proyectiva de las cuatro cámaras es inmediata. En la literatura no se ha encontrado ningún artículo que extraiga las cuatro matrices de proyección dado un tensor cuadrifocal arbitrario.

6.5.2.4. Estimación iterativa

El algoritmo de la última sección proporciona una forma de calcular un vector de error algebraico $\epsilon = \hat{\mathbf{A}}\mathbf{q}$ suponiendo que el vector $\mathbf{a} \in \mathbb{R}^9$ es conocido. Encontramos una aplicación como la siguiente:

$$\begin{array}{rccc} f: & \mathbb{R}^9 & \longrightarrow & \mathbb{R}^{81} \\ & \mathbf{a} & \mapsto & \epsilon = \hat{\mathbf{A}}\mathbf{E}\mathbf{M}\mathbf{a}' = \hat{\mathbf{A}}\mathbf{q} \end{array}$$

Esto sugiere variar $\mathbf{a}$ de forma que se minimice el error algebraico $\|\hat{\mathbf{A}}\mathbf{q}\|$, para lo cual f puede utilizarse como la función modelo en un esquema iterativo de mínimos cuadrados no lineales, a resolver mediante el algoritmo de Levenberg-Marquardt.

Uno puede observar que la solución obtenida de esta forma es un mínimo sujeto a que las transformaciones $\mathtt{T}, \ldots, \mathtt{T}'''$ son fijas. Estas transformaciones se obtienen de las coordenadas de las tres primeras correspondencias de puntos. Podríamos dejar que variasen los elementos de las transformaciones $\mathtt{T}', \mathtt{T}'', \mathtt{T}'''$ para encontrar el mínimo absoluto del error algebraico. No hace falta permitir que la transformación $\mathtt{T}$ varíe. Como $\mathtt{T}', \mathtt{T}'', \mathtt{T}'''$ están determinadas por las coordenadas $\mathbf{u}'_i, \mathbf{u}''_i, \mathbf{u}'''_i$ para $i = 1, \ldots, 3$, esto añade parámetros variables a la estimación, un total de $9 + 6 \times 3 = 27$. Así, pues quedaría una aplicación

$$\begin{array}{rccc} f: & \mathbb{R}^{27} & \longrightarrow & \mathbb{R}^{81} \\ & (\mathbf{a}, \mathbf{u}_{a1}, \mathbf{u}'_{a1}, \mathbf{u}''_{a1}, \mathbf{u}_{a2}, \mathbf{u}'_{a2}, \mathbf{u}''_{a2}, \mathbf{u}_{a3}, \mathbf{u}'_{a3}, \mathbf{u}''_{a3}) & \mapsto & \epsilon = \hat{\mathbf{A}}\mathbf{E}\mathbf{M}\mathbf{a}' = \hat{\mathbf{A}}\mathbf{q} \end{array}$$

siendo $\mathbf{u}_{ai}$ coordenadas afines 2×1.

6.5.2.5. Minimización del error de reproyección

Según la suposición habitual de que el error en las medidas sobre la imagen siguen una distribución gaussiana, dado un conjunto de correspondencias medidas $\mathbf{u}_i \leftrightarrow \mathbf{u}'_i \leftrightarrow \mathbf{u}''_i \leftrightarrow \mathbf{u}'''_i$, la estimación óptima (máxima verosimilitud) del tensor cuadrifocal bajo este modelo de error es la que satisface ecuaciones de la forma

$$\hat{u}^i \hat{u}'^j \hat{u}''^k \hat{u}'''^l \epsilon_{ipw} \epsilon_{jqx} \epsilon_{kry} \epsilon_{lsz} Q^{pqrs} = 0_{wxyz} \tag{6.24}$$

para cada correspondencia de puntos, donde $\hat{\mathbf{u}}_i, \hat{\mathbf{u}}'_i, \hat{\mathbf{u}}''_i$ y $\hat{\mathbf{u}}'''_i$ son puntos estimados en las imágenes que minimizan el error geométrico

$$\sum_i \left[d(\mathbf{u}_i, \hat{\mathbf{u}}_i)^2 + d(\mathbf{u}'_i, \hat{\mathbf{u}}'_i)^2 + d(\mathbf{u}''_i, \hat{\mathbf{u}}''_i)^2 + d(\mathbf{u}'''_i, \hat{\mathbf{u}}'''_i)^2 \right] \tag{6.25}$$

y verifican de forma exacta (6.24).

El error geométrico se puede minimizar utilizando la técnica del ajuste de haces, que se verá más adelante. En general, cabe esperar que la minimización del error geométrico proporcione los mejores resultados posibles, dependiendo de cuan realista sea el modelo del ruido. Más adelante se darán los límites teóricos de este algoritmo de máxima verosimilitud.

6.6. Geometría multicámara

Las siguientes secciones están dedicadas a los algoritmos de calibración en caso de haber más de dos o tres cámaras. Se trata de calcular la posición de un punto en el espacio 3D dadas sus proyecciones en varias imágenes, a la vez que estimar las matrices de proyección de dichas cámaras. Una posible guía referencia es el capítulo 17 de [1].

Antes de nada se cuentan el número de ecuaciones y de incógnitas que rigen la calibración proyectiva de varias cámaras, de forma similar a como se hizo en la formalización del problema, § 2.3, sólo que

entonces se hizo desde un enfoque euclídeo, no proyectivo.

La sección de algoritmos está organizada de la siguiente forma: primero se estima una calibración proyectiva inicial, para después optimizarla, operación conocida como *Ajuste de Haces (Bundle Adjustment)*. Después se aplican los conocimientos del RANSAC para crear un ajuste de haces proyectivo robusto. *Por último*, se complica el modelo introduciendo la estimación de la distorsión radial de las cámaras.

Objetivos: Estimar una reconstrucción o calibración proyectiva óptima. Optimizar una calibración proyectiva en condiciones de distorsión radial.

6.6.1. Resumen numérico

Recopilemos algunos datos de secciones anteriores sobre el número de puntos o rectas que hacen falta para obtener una calibración proyectiva de varias imágenes. El análisis se basa en contar los grados de libertad de los tensores utilizados. La matriz fundamental también se puede interpretar como un tensor (el tensor bifocal o epipolar), pero en la literatura es más popular la formulación matricial.

Empecemos con la matriz fundamental: es una matriz homogénea de 3×3, por lo tanto tiene $9 - 1$ grados de libertad, pero debe satisfacer la restricción $\det \mathtt{F} = 0$, por lo que resultan $9-1-1 = 7$ grados de libertad. Otra forma de calcularlo: una matriz fundamental proviene de dos matrices de proyección ($(12 - 1) \cdot 2 = 22$ grados de libertad), pero cualquier calibración proyectiva está determinada salvo una homografía arbitraria del espacio (15 grados de libertad). Así que en total cuentan $11 \cdot 2 - 15 = 7$ grados de libertad.

El tensor trifocal codifica la estructura proyectiva de 3 matrices de proyección, así que tiene $11*3-15 = 18$ grados de libertad. Del mismo modo, el tensor cuadrifocal posee $11 \cdot 4 - 15 = 29$ grados de libertad. En general, para m cámaras hay $11m - 15$ grados de libertad.

Los tensores trifocal y cuadrifocal tienen, respectivamente, $3^3 = 27$ y $3^4 = 81$ elementos, salvo proporcionalidad. Además acabamos de calcular el número de grados de libertad que poseen, así que podemos calcular en número de restricciones algebraicas que deben satisfacer: $(27 - 1) - 18 = 8$ el tensor trifocal y $(81 - 1) - 29 = 51$ el tensor cuadrifocal.

Consideremos una configuración de n puntos y m cámaras. El número total de grados de libertad del sistema es $11m - 15 + 3n$, ya que cada punto 3D tiene 3 grados de libertad. Los datos para estimar la estructura proyectiva de la escena son los n puntos observados en las m imágenes, un total de $2mn$ medidas porque cada punto 2D tiene 2 coordenadas. Así pues, para que la reconstrucción sea posible se debe cumplir $2mn \geq 11m - 15 + 3n$, o $(2m - 3)n \geq 11m - 15$. El número de puntos necesarios es

$$n \geq \frac{11m - 15}{2m - 3} = 5 + \frac{m}{2m - 3}.$$

Hay una cota inferior de $5\frac{1}{2}$ (6 porque son enteras) correspondencias de puntos. Un argumento similar se utiliza en caso de rectas, cada una con 4 grados de libertad en el espacio $\mathbb{P}^3$ (6 coordenadas de Plücker -1 factor de escala -1 restricción de pertenencia a la cuádrica de Klein); ó 2 grados de libertad en el plano. Así que $2mn \geq 11m - 15 + 4n$, es decir

$$n \geq \frac{11m - 15}{2m - 4}$$

La tabla 6.3 resume el número de correspondencias necesarias para la reconstrucción proyectiva de la escena. La columna "algoritmo lineal" indica el número de correspondencias de puntos o rectas necesarias para calcular el tensor, salvo factor de escala. Son los algoritmos de las secciones § 6.2.3.2, § 6.4.3.1 y § 6.5.2.1. La columna de los algoritmos no-lineales indica el número mínimo de correspondencias para estimar el tensor. Si hay un asterisco, es que son posibles varias soluciones, como son

m	tensor	Número elementos	Grados libertad	algoritmo lineal		algoritmo no lineal	
				puntos	rectas	puntos	rectas
2	F	9	7	8	-	7*	-
3	$\mathcal{T}$	27	18	7	13	6*	9?
4	Q	81	29	6	9	6	8?

Cuadro 6.3: Grados de libertad y restricciones en la calibración proyectiva

los algoritmos de § 6.2.3.1 y § 6.4.3.4. Una interrogación indica que no se conoce ningún algoritmo práctico para esa situación.

6.6.2. Planteamiento del problema

Consideremos la situación en la que un conjunto de puntos 3D $\mathbf{X}_j$ son visualizados por un conjunto de cámaras con matrices de proyección $\mathtt{P}^i$. Llamemos $\mathbf{x}^i_j$ a las coordenadas de la proyección del j-ésimo punto en la i-ésima cámara. Queremos resolver el siguiente problema de reconstrucción: dado un conjunto de puntos proyectados $\mathbf{x}^i_j$, encontrar el conjunto de matrices de proyección de las cámaras, $\mathtt{P}^i$, y los puntos $\mathbf{X}_j$ tales que $\mathtt{P}^i\mathbf{X}_j = \mathbf{x}^i_j$. Sin más restricciones en las matrices de proyección ni en los puntos 3D, tal reconstrucción se conoce como reconstrucción proyectiva, porque los puntos $\mathbf{X}_j$ se diferencian de la reconstrucción verdadera (euclídea o métrica) en una transformación proyectiva del espacio arbitraria.

Ajuste de haces

Si las medidas en las imágenes son ruidosas, entonces las ecuaciones $\mathtt{P}^i\mathbf{X}_j = \mathbf{x}^i_j$ no se cumplirán exactamente. En este caso se busca la solución de Máxima Verosimilitud (ML) suponiendo que el ruido es gaussiano: queremos estimar las m matrices de proyección $\hat{\mathtt{P}}^i$, y los n puntos 3D $\hat{\mathbf{X}}_j$ que se proyectan exactamente en los puntos $\hat{\mathbf{x}}^i_j$ que $\hat{\mathbf{x}}^i_j = \hat{\mathtt{P}}^i\hat{\mathbf{X}}_j$, y a la vez minimizan la distancia entre el punto reproyectado y el punto observado (medido) $\mathbf{x}^i_j$, para cada imagen en la que se proyecta el punto 3D, esto es:

$$\min_{\hat{\mathtt{P}}^i, \hat{\mathbf{X}}_j} \sum_{i=1}^{m}\sum_{j=1}^{n} d(\hat{\mathtt{P}}^i\hat{\mathbf{X}}_j, \mathbf{x}^i_j)^2 \tag{6.26}$$

donde $d(\mathbf{x}, \mathbf{y})$ es la distancia euclídea en el plano de la imagen entre los puntos en coordenadas homogéneas $\mathbf{x}$ e $\mathbf{y}$. Esta parametrización, que implica la minimización del error de reproyección, se conoce como *Ajuste de Haces* (*Bundle Adjustment*) porque implica ajustar los haces de rayos entre cada centro óptico de cada cámara y el conjunto de puntos 3D.

El ajuste de haces debería, en general, ser utilizado como el paso final de cualquier algoritmo de reconstrucción. Este método tiene la ventaja de ser tolerante si se pierden datos a la vez que proporciona una auténtica estimación de Máxima Verosimilitud. El algoritmo tiene dos inconvenientes: (i) necesita una buena inicialización y (ii) puede llegar a ser un problema de minimización de grandes dimensiones del espacio de parámetros.

6.6.3. Reconstrucción inicial

Es la fase de preparación de la calibración proyectiva, para su posterior optimización. El objetivo es obtener una aproximación inicial a las matrices de proyección $\mathtt{P}^i$ de tal forma que todas estén expresadas en una misma referencia proyectiva espacial $\mathcal{R}$: sólo hay un conjunto de puntos 3D $\mathbf{X}_j$.

Existen multitud de posibilidades, que listamos a continuación. Supongamos que las imágenes pertenecen a una secuencia de vídeo, en este caso cabe esperar que a medida que avance la secuencia, las imágenes

presenten cada vez puntos de vistas más diferentes respecto de la primera imagen: es lo que a veces se llama: "suficiente *baseline* (distancia entre las posiciones de los centros ópticos en el mundo real) entre las imágenes".

1. Estimar la matriz fundamental entre la primera y la última imágenes, ya que es de esperar que sean las que más separadas están en el espacio. Así lograremos que la triangulación para obtener los puntos 3D tenga menos incertidumbre (se comentan menos errores). Estimar las matrices de las cámaras intermedias mediante lo que se conoce como *resectioning*, que lo traduciremos por resección, ya que no se ha encontrado un término equivalente en el DRAE. La resección consiste en el cálculo de la matriz de proyección si son conocidas las posiciones de los puntos 3D y los puntos en la imagen.

   Esta estrategia es bastante buena y hay varias combinaciones: ya que existen tanto algoritmos lineales como óptimos (Gold Standard) para los tres pasos principales: estimación de la matriz fundamental y matrices de proyección de las cámaras extremas, triangulación y resección.

2. Estimar el tensor trifocal entre la primera imagen, la última y una intermedia. Estimar las matrices del resto de cámaras mediante resección.

3. Estimar el tensor cuadrifocal entre cuatro imágenes equiespaciadas incluyendo las dos extremas. Estimar las matrices del resto de cámaras mediante resección.

4. Obtener las matrices de proyección a partir de la matriz fundamental (según § 6.2.5.2), calculada de 2 en 2 imágenes. Sin embargo, esto no garantiza que las matrices de proyección estén expresadas en la misma referencia espacial $\mathcal{R}$, para conseguirlo hay que realizar cambios de referencia.

5. Obtener las matrices de proyección a partir del tensor trifocal, calculado de 3 en 3 imágenes. Esto tampoco garantiza que las matrices de proyección estén expresadas en la misma referencia espacial $\mathcal{R}$ y también hay que realizar cambios de referencia.

6. La misma idea que el método anterior, pero de 4 en 4 imágenes con el tensor cuadrifocal.

Vemos que entre las opciones no se encuentra el método de Peter Sturm de la factorización proyectiva que se indica en [1]. A mi juicio y según varias simulaciones realizadas es un algoritmo iterativo que no minimiza explícitamente ninguna función de coste y sin garantías de convergencia.

Cambio de referencia espacial

Comentemos los cambios de referencia de los tres últimos métodos. Un cambio de referencia de matriz $\mathtt{C}$, está proyectivamente justificado porque

$$\lambda\mathbf{x} = \mathtt{P}\mathbf{X} = (\mathtt{PC})(\mathtt{C}^{-1}\mathbf{X}) = \tilde{\mathtt{P}}\tilde{\mathbf{X}}$$

Si los puntos 3D se transforman de acuerdo a $\tilde{\mathbf{X}} = \mathtt{C}^{-1}\mathbf{X}$, las matrices de proyección lo hacen según $\tilde{\mathtt{P}} = \mathtt{PC}$.

Cada vez que se realiza la calibración proyectiva de dos cámaras a partir de unas correspondencias de puntos $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ se obtienen dos matrices de proyección $\mathtt{P}, \mathtt{P}'$ y los puntos 3D $\mathbf{X}$ que mejor se proyectan en $\mathbf{x}_i, \mathbf{x}'_i$. Si hay más de dos cámaras, hay que conseguir que todos los $\mathbf{X}$ sean "iguales" (los nombrados $\tilde{\mathbf{X}}$) y en consecuencia, modificar $\mathtt{P}, \mathtt{P}'$.

Los principales pasos del método 4 se pueden resumir en: dadas n correspondencias $\mathbf{x}^i_j$ entre m imágenes,

1. Elegir como referencia destino $(\mathcal{R}, \tilde{\mathbf{X}})$, una referencia proyectiva aleatoria cuyos vectores de la base asociada sean ortogonales (las columnas de una matriz ortogonal cualquiera).

2. Para cada dos imágenes:

   a) **Calibración proyectiva de dos cámaras**:

      1) Hallar la matriz fundamental entre ambas imágenes: $\mathtt{F}_{i,i+1}$.
      2) Obtener las dos matrices de proyección a partir de $\mathtt{F}_{i,i+1}$ y los puntos 3D correspondientes $\mathbf{X}$ en una referencia $\mathcal{R}^*$. La primera matriz tiene siempre la forma: $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$.
      3) Si es la calibración de las dos primeras cámaras, se eligen los índices de los 5 puntos de las referencias proyectivas origen $\mathbf{X}$.

   b) Obtener la matriz $\mathtt{C}$ del cambio de referencia que transforma los 5 puntos de la referencia origen $(\mathcal{R}^*, \mathbf{X})$ en los puntos $(\mathcal{R}, \tilde{\mathbf{X}})$.

   c) **Corrección**:

      1) Transformar ambas matrices de proyección: $\tilde{\mathtt{P}} = \mathtt{PC}$ y $\tilde{\mathtt{P}}' = \mathtt{P}'\mathtt{C}$.
      2) Transformar los puntos 3D: $\tilde{\mathbf{X}}' = \mathtt{C}^{-1}\mathbf{X}$

Comparación de las dos familias

En la lista de métodos anteriores podemos hacer una clara distinción entre los tres primeros y los tres últimos. Hay dos problemas en los tres últimos métodos: uno potencialmente grave y otro no tanto.

El problema grave es el paso crítico del cambio de referencia, que para serlo debe estar definido por puntos que formen una verdadera referencia. En el caso de $\mathbb{P}^3$, la referencia está formada por cinco puntos y deben ser linealmente independientes cualesquiera cuatro de ellos. De no ser así, la matriz del cambio no es de rango máximo, sino que es degenerada y el cambio de referencia no pude recibir tal nombre: está mal hecho porque lleva todos los puntos del espacio a un plano, a lo sumo.

En la práctica, no siempre es fácil de estimar la independencia. Numéricamente no basta que sean independientes, deben ser "lo más independientes posible"; sería deseable tener la mejor referencia: el tetraedro regular y su baricentro, mas casi nunca es posible. Además, los datos de partida son las observaciones: los puntos en las imágenes, y no parece trivial determinar si las proyecciones de cinco puntos en varias imágenes forman una buena referencia proyectiva.

Otro inconveniente es que de todos los puntos sólo intervienen cinco en la elección el cambio de referencia. Es verdad que un correcto cambio de referencia hace que las coordenadas homogéneas espaciales de esos cinco puntos coincidan, mas se está dando preferencia a esos cinco respecto del resto. En presencia de ruido se ha comprobado que pueden existir grandes diferencias entre las coordenadas homogéneas del resto de puntos, comparando en la misma referencia los obtenidos con varios cambios, lo que da lugar a grandes costes del error de reproyección, que es la forma de medir la bondad de estos cambios.

La primera familia presenta, en cambio, un mejor comportamiento. Por ejemplo, en el primer método sólo hay que estimar bien las matrices de proyección de dos cámaras y las coordenadas de los puntos 3D, esto se pude hacer mediante el Gold Standard para la matriz fundamental. El resto de matrices de proyección se pueden obtener minimizando el error de reproyección entre los puntos 3D proyectados y las observaciones. Si en lugar de esto se elige un algoritmo lineal, al menos esa solución mínimo-cuadrática tampoco da preferencia a ningún punto sobre otros.

Este paso de obtención de la calibración proyectiva inicial tiene que ser lo suficientemente bueno como para que el ajuste de haces posterior tenga una optimización fácil. Como en todos los algoritmos de optimización, es deseable que el punto de partida caiga dentro de la zona de atracción del mínimo. Sin embargo no debe ser demasiado costoso obtener el punto de partida, ya que la optimización posterior puede ser más rápida. Según este criterio, los métodos basados en los tensores trifocal y cuadrifocal son más lentos que los basados en la matriz fundamental y no ofrecen significativamente mejores resultados

como punto de partida. Después de todas estas discusiones, concluimos que el mejor método para una reconstrucción proyectiva inicial es el primero.

6.6.4. Ajuste de Haces Proyectivo

Una vez que se dispone de una calibración proyectiva inicial, es posible realizar una optimización para minimizar el error de reproyección de la ecuación (6.26), ya que el hecho de realizar la calibración proyectiva por cualquiera de los métodos comentados en la sección anterior no asegura que (6.26) se minimice.

Para la optimización utilizamos el algoritmo de Levenberg-Marquardt y el marco descriptivo común, según se indica en § A4.6 de [1].

Lo diferente de este problema respecto a los anteriores es que el vector de parámetros $\mathbf{P}$ no sólo tiene particionada la parte $\mathbf{b}$ (coordenadas afines de los puntos 3D), sino también la parte $\mathbf{a}$ (m matrices de proyección). Por lo tanto, hay más condiciones de dispersión (o independencia). Así que la matriz jacobiana $\mathtt{J} = [\mathtt{A} \,|\, \mathtt{B}]$ tiene la parte $\mathtt{B}$ diagonal a bloques y además cada matriz $\mathtt{A}_i$ es diagonal a bloques. Las matrices $\mathtt{U}, \mathtt{V}$ de $\mathtt{J}^\top\mathtt{J}$ son diagonales a bloques también: $\mathtt{U}$ tiene tantos bloques como cámaras (imágenes) y $\mathtt{V}$ tiene tantos como puntos por imagen, como se aprecia en la figura 6.11.

Funciones modelo y de coste
Veamos las funciones de diseño del problema en términos de optimización LM. La función modelo es:

$$f : \left[\frac{\mathbf{a}}{\mathbf{b}}\right] = \left[\begin{array}{c} \mathtt{P}^1 \\ \vdots \\ \mathtt{P}^m \\ \hline \hat{\mathbf{X}}_{a1} \\ \vdots \\ \hat{\mathbf{X}}_{a\,n} \end{array}\right] \equiv \mathbf{P} \rightarrow \widehat{\mathbf{X}} \equiv \left[\begin{array}{c} \left(\begin{array}{c} \hat{\mathbf{x}}^1_{a1} \\ \vdots \\ \hat{\mathbf{x}}^m_{a1} \end{array}\right) \\ \vdots \\ \left(\begin{array}{c} \hat{\mathbf{x}}^1_{a\,n} \\ \vdots \\ \hat{\mathbf{x}}^m_{a\,n} \end{array}\right) \end{array}\right] = \left[\begin{array}{c} \widehat{\mathbf{X}}_{a1} \\ \vdots \\ \widehat{\mathbf{X}}_{a\,n} \end{array}\right]$$

El vector de parámetros $\mathbf{P}$ de $M = 12m + 3n$ componentes puede ser considerablemente grande. El vector de medidas $\widehat{\mathbf{X}}$ está formado por las coordenadas afines de los puntos proyectados $\mathbf{x}^i_j$ y es de dimensión $N = 2mn$. La función de coste es el error de reproyección de los n puntos en las m imágenes, es decir,

$$\text{coste} = \|\mathbf{X} - \widehat{\mathbf{X}}\| = \|\mathbf{X} - f(\mathbf{P})\| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} d(\hat{\mathtt{P}}^i\hat{\mathbf{X}}_j, \mathbf{x}^i_j)^2}$$

El algoritmo de Levenberg-Marquardt particionado disperso adaptado al ajuste de haces está explicado en [1, pág. 582]. La matriz jacobiana se debe construir de forma exacta a partir de los datos, no numéricamente porque sería mucho más lento. Posee una estructura dispersa como la mostrada en la figura 6.11. Por lo tanto, la matriz $\mathtt{J}^\top\mathtt{J}$ también tiene estructura dispersa: las m submatrices $\mathtt{U}_j$ que componen la matriz $\mathtt{U}$ diagonal a bloques; las n submatrices $\mathtt{V}_i$ que forman la matriz $\mathtt{V}$ diagonal a bloques, y las submatrices $\mathtt{W}$ y $\mathtt{W}^\top$.

$$\mathtt{J} = \begin{pmatrix} \mathtt{A} & \mathtt{B} \end{pmatrix} \qquad\qquad \mathtt{J}^\top\mathtt{J} = \begin{pmatrix} \mathtt{U} & \mathtt{W} \\ \mathtt{W}^\top & \mathtt{V} \end{pmatrix}$$

Las matrices jacobianas $\mathtt{A}_{ij}(2\times 9)$ y $\mathtt{B}_{ij}(2\times 3)$ tienen una forma conocida. Siendo

$$\hat{\mathbf{x}}^j_{ai} = \left(\frac{x^j_i}{w^j_i}, \frac{y^j_i}{w^j_i}\right)^\top,$$

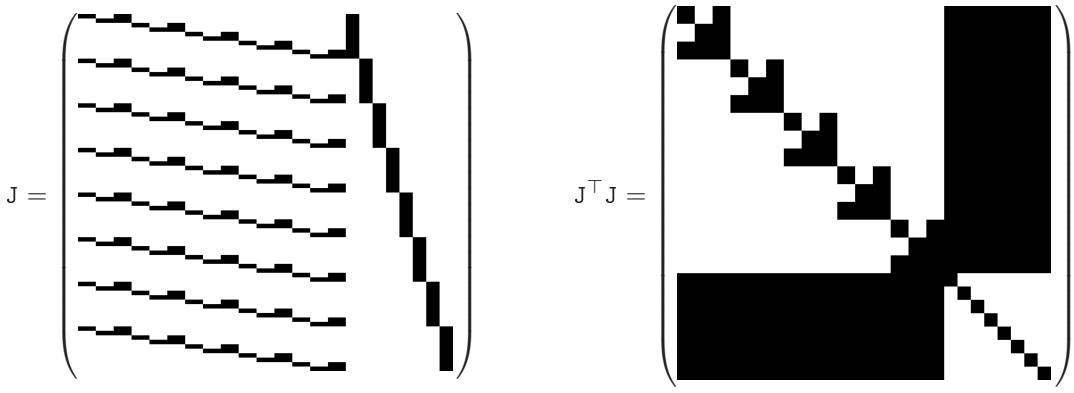


Figura 6.11: Estructura de las matrices jacobiana ($\mathtt{J}$) y $\mathtt{J}^\top\mathtt{J}$ del ajuste de haces proyectivo, con $m = 5$ cámaras y $n = 8$ puntos

las ecuaciones de las derivadas exactas son:

$$\mathtt{A}_{ij} = \left[\frac{\partial \hat{\mathbf{x}}_{ai}^j}{\partial \mathbf{a}_j}\right] = \left[\frac{\partial \hat{\mathbf{x}}_{ai}^j}{\partial \mathtt{P}^j}\right] = \frac{1}{w_i^j}\begin{bmatrix} \hat{\mathbf{X}}_{hi}^\top & \mathbf{0}^\top & -\dfrac{x_i^j}{w_i^j}\hat{\mathbf{X}}_{hi}^\top \\ \mathbf{0}^\top & \hat{\mathbf{X}}_{hi}^\top & -\dfrac{y_i^j}{w_i^j}\hat{\mathbf{X}}_{hi}^\top \end{bmatrix} = \begin{bmatrix} \mathtt{I} & -\hat{\mathbf{x}}_{ai}^j \end{bmatrix} \otimes \frac{1}{w_i^j}\hat{\mathbf{X}}_{hi}^\top$$

donde $\hat{\mathbf{X}}_{hi}^\top = (\hat{\mathbf{X}}_{ai}^\top, 1)$ son las coordenadas homogéneas del i-ésimo punto 3D del vector de parámetros.

$$\mathtt{B}_{ij} = \left[\frac{\partial \hat{\mathbf{x}}_{ai}^j}{\partial \mathbf{b}_i}\right] = \left[\frac{\partial \hat{\mathbf{x}}_{ai}^j}{\partial \hat{\mathbf{X}}_{ai}}\right] = \frac{1}{w_i^j}\left(\begin{bmatrix} \mathtt{P}_{11}^j & \mathtt{P}_{12}^j & \mathtt{P}_{13}^j \\ \mathtt{P}_{21}^j & \mathtt{P}_{22}^j & \mathtt{P}_{23}^j \end{bmatrix} - \hat{\mathbf{x}}_{ai}^j[\mathtt{P}_{31}^j\ \mathtt{P}_{32}^j\ \mathtt{P}_{33}^j]\right)$$

6.6.4.1. Estimación robusta mediante RANSAC

Si la calibración proyectiva inicial no es buena, es muy costoso realizar una ajuste de haces con todos los puntos y cámaras desde un principio. Por eso se ha ideado una estrategia basada en el RANSAC para mejorar la eficacia.

El algoritmo está resumido en los siguientes pasos: dadas las n correspondencias de puntos entre las m imágenes $\mathbf{x}_j^i$:

1. **RANSAC básico**: Repetir hasta haber probado N muestras (donde N es determinado adaptativamente según el algoritmo 3.5, [1, pág. 105]) o haber logrado más de $T = (1 - \epsilon_{\text{fijo}})n$ puntos consistentes con el modelo (*inliers*).

   a) Seleccionar una muestra aleatoria de 8 correspondencias de puntos y realizar un ajuste de haces sobre estos puntos en todas las cámaras.

   b) Calcular la distancia $d_\perp$ de cada punto al modelo de proyección indicado por el ajuste de haces.

   c) Calcular el número de puntos consistentes con el modelo como el número de correspondencias de puntos para los que $d_\perp < t = k_{th}\sigma$ píxeles, siendo σ la desviación típica del ruido gaussiano esperado.

   d) Comparar: Retener la calibración proyectiva de la muestra con el mayor soporte (mayor número de *inliers*). En el caso de empate, retener la solución con la menor desviación típica de *inliers* (en cuanto a distancia al modelo $d_\perp$).

   e) Estimar la proporción ϵ de puntos no consistentes (*outliers*) y el número de muestras aleatorias a probar, N.

2. **Estimación óptima no lineal**: realizar un ajuste de haces minimizando el error de reproyección a partir del mayor y mejor soporte encontrado, sólo sobre las correspondencias clasificadas como consistentes con el modelo.

3. **Identificación de nuevas parejas** consistentes con el modelo, igual que en los pasos c) y d). Continuar minimizando la distancia geométrica (paso 2) hasta que el soporte deje de crecer.

Este algoritmo tiene la ventaja de que realizar el ajuste de haces sobre los puntos de una muestra es mucho más rápido que sobre todos los puntos y esta muestra sirve para poner coordenadas espaciales al resto de puntos proyectados, una vez más mediante triangulación.

Umbral de clasificación

Obtengamos el valor teórico del umbral de clasificación de los puntos, para el modelo de ruido gaussiano habitual. Cada punto se clasifica según una distancia al modelo $d^2_{\perp j} = \sum_{i=1}^{m} d(\mathbf{x}^i_j, \hat{\mathtt{P}}^i \hat{\mathbf{X}}_j)^2$. Ésta es la cantidad que se debe comparar con un umbral. Veamos qué distribución sigue.

La distancia la podemos obtener como la norma del vector

$$\mathbf{g} = \mathbf{x}_j - \hat{\mathbf{x}}_j = [x^1_j - \hat{x}^1_j, y^1_j - \hat{y}^1_j, \ldots, x^m_j - \hat{x}^m_j, y^m_j - \hat{y}^m_j]^\top$$

de dimensión $2m$. Cada coordenada es una variable aleatoria independiente de varianza σ^2, por hipótesis. Recordemos que una variable χ^2 se obtiene como suma de cuadrados de variables aleatorias gaussianas normalizadas (de media cero y $\sigma = 1$). Para un vector $\mathbf{g}$ como el anterior, la cantidad $\|\mathbf{g}\|^2/\sigma^2$ sigue una distribución χ^2_{2m} con $2m$ grados de libertad ya que todas las componentes de $\mathbf{g}$ son independientes. Para esta variable χ^2 podemos hallar el valor para el cual la función de distribución acumulada alcanza el 95 % de su máximo: $k^2_{th} = F^{-1}_{2m}(\alpha = 0{,}95) = \|\mathbf{g}\|^2/\sigma^2$. Así que

$$d^2_{\perp\alpha} = \|\mathbf{g}_\alpha\|^2 = F^{-1}_{2m}(0{,}95)\sigma^2$$

Como se aprecia, el umbral de clasificación $d_{\perp\alpha}$ es mayor cuanto mayor número de cámaras, pero no sigue una ley lineal.

6.6.5. Ajuste de Haces Proyectivo con Distorsión Radial

En esta sección se añade al modelo proyectivo lineal un modelo de distorsión radial de las lentes de las cámaras. La forma cómoda de incluir este modelo en un ajuste de haces proyectivo es distorsionar las coordenadas afines proyectadas. Como ya se ha comentado en § 2.2.4, algunos autores incluyen el modelo en medio de la proyección, antes de aplicar la matriz de parámetros intrínsecos, es decir, modifican las coordenadas normalizadas que se usarían para estimar la matriz esencial. En nuestro caso, como suponemos que las cámaras no están calibradas, desconocemos `K`, lo único que se puede hacer es modificar las coordenadas de los puntos proyectados por la matriz `P`. Más referencias: [41], [42].

La función de coste a minimizar es el error de reproyección de los puntos proyectados y distorsionados $\hat{\mathbf{x}}^i_{dj}$ respecto de los puntos medidos en las imágenes $\mathbf{x}^i_j$ (los datos).

$$\min_{\hat{\mathtt{P}}^i, \boldsymbol{\kappa}^i, \hat{\mathbf{X}}_j} \sum_{i=1}^{m} \sum_{j=1}^{n} d(\hat{\mathbf{x}}^i_{dj}, \mathbf{x}^i_j)^2 \tag{6.27}$$

El modelo de distorsión radial que se ha escogido es el que se ha presentado en § 2.2.4 e indica cómo obtener las estimaciones $\hat{\mathbf{x}}^i_{dj}$. Supongamos que conocemos los puntos 3D y las matrices de proyección a la vez que los parámetros de distorsión radial de cada cámara, $\boldsymbol{\kappa}^i$. Proyectamos linealmente los puntos y obtenemos

$$\hat{\mathbf{x}}^i_j = \hat{\mathtt{P}}^i \hat{\mathbf{X}}_j = (x^i_j, y^i_j, w^i_j)^\top$$

A continuación deshomogeneizamos los $\hat{\mathbf{x}}^i_j$, para obtener coordenadas afines.

$$\hat{\mathbf{x}}^i_{aj} = \left(\frac{x^i_j}{w^i_j}, \frac{y^i_j}{w^i_j}\right)^\top = (x^i_{aj}, y^i_{aj})^\top$$

A partir de los parámetros $\boldsymbol{\kappa}^i = (x^i_c, y^i_c, \kappa_1, \ldots, \kappa_{\text{ncoef}})^\top$ creamos el polinomio radial que distorsiona las coordenadas afines, desde el centro de distorsión $\hat{\mathbf{x}}^i_{ac} = (x^i_c, y^i_c)^\top$.

$$L(r) = 1 + \sum_{m=1}^{\text{ncoef}} \kappa_m r^m \tag{6.28}$$

Y las coordenadas distorsionadas son:

$$\hat{\mathbf{x}}^i_{adj} = \hat{\mathbf{x}}^i_{ac} + L(r^i_j)(\hat{\mathbf{x}}^i_{aj} - \hat{\mathbf{x}}^i_{ac}) = \begin{pmatrix} x^i_c \\ y^i_c \end{pmatrix} + L(r^i_j) \begin{pmatrix} x^i_{aj} - x^i_c \\ y^i_{aj} - y^i_c \end{pmatrix} \tag{6.29}$$

En coordenadas homogéneas,

$$\hat{\mathbf{x}}^i_{dj} = (\hat{\mathbf{x}}^{i\top}_{aj}, 1)^\top \tag{6.30}$$

siendo r^i_j el radio de cada punto proyectado linealmente al centro de la distorsión radial.

$$(r^i_j)^2 = (x^i_{aj} - x^i_c)^2 + (y^i_{aj} - y^i_c)^2$$

El valor de ncoef se puede escoger entre 0 y cualquier número natural, pero como mucho se suele incluir hasta el término de cuarto orden. Por convenio, si ncoef $= 0$, no hay distorsión radial porque se elimina el sumatorio en (6.28). Así que lo normal es elegir ncoef entre 1 y 4. El polinomio distorsionador quedaría

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \kappa_4 r^4 \tag{6.31}$$

Una vez que el modelo es conocido y está justificado, se entiende el enunciado del problema: estimar las matrices de proyección $\hat{\mathtt{P}}^i$ junto con sus parámetros de distorsión radial $\boldsymbol{\kappa}^i$ y los puntos 3D $\hat{\mathbf{X}}_j$, que se proyectan exactamente en los puntos $\hat{\mathbf{x}}^i_{dj}$ (6.30) y a la vez minimizan (6.27).

La modificación que hay que hacer al ajuste de haces proyectivo de § 6.6.4 es incluir los nuevos parámetros de distorsión radial de las matrices de proyección en el vector de parámetros de la función modelo.

Funciones modelo y de coste

Veamos las funciones de diseño del problema. Función modelo:

$$f : \begin{bmatrix} \mathbf{a} \\ \hline \mathbf{b} \end{bmatrix} = \left[\begin{array}{c} \begin{pmatrix} \mathtt{P}^1 \\ \boldsymbol{\kappa}^1 \end{pmatrix} \\ \vdots \\ \begin{pmatrix} \mathtt{P}^m \\ \boldsymbol{\kappa}^m \end{pmatrix} \\ \hline \hat{\mathbf{X}}_{a1} \\ \vdots \\ \hat{\mathbf{X}}_{a\,n} \end{array}\right] \equiv \mathbf{P} \to \widehat{\mathbf{X}} \equiv \begin{bmatrix} \begin{pmatrix} \hat{\mathbf{x}}^1_{ad1} \\ \vdots \\ \hat{\mathbf{x}}^m_{ad1} \end{pmatrix} \\ \vdots \\ \begin{pmatrix} \hat{\mathbf{x}}^1_{ad\,n} \\ \vdots \\ \hat{\mathbf{x}}^m_{ad\,n} \end{pmatrix} \end{bmatrix} = \begin{bmatrix} \widehat{\mathbf{X}}_{ad1} \\ \vdots \\ \widehat{\mathbf{X}}_{ad\,n} \end{bmatrix}$$

El vector de parámetros $\mathbf{P}$ tiene dimensión $M = (12 + 2 + \text{ncoef})m + 3n$. El vector de medidas $\widehat{\mathbf{X}}$, dimensión $N = 2mn$ y se forma con las coordenadas afines distorsionadas, $\hat{\mathbf{x}}^i_{adj}$. La función de coste es el error de reproyección en coordenadas distorsionadas.

$$\text{coste} = \|\mathbf{X} - \widehat{\mathbf{X}}\| = \|\mathbf{X} - f(\mathbf{P})\| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} d(\hat{\mathbf{x}}^i_{dj}, \mathbf{x}^i_j)^2}$$

La matriz jacobiana de este problema tiene un número de columnas variable, dependiendo del número de coeficientes de distorsión radial que se especifiquen en ncoef. Es también dispersa, con la misma estructura que la de una matriz de ajuste de haces genérica (figura 6.11). Las matrices $\mathtt{A}_{ij}$ son de $2 \times (12 + 2 + \text{ncoef})$ y las matrices $\mathtt{B}_{ij}$ mantienen el tamaño 2×3. Como se ha cambiado la parte **a** del vector de parámetros, sólo cambia la parte `A` de la matriz jacobiana.

Inicialización

Como punto de partida de la búsqueda se pueden utilizar dos alternativas: una calibración proyectiva inicial o una calibración proyectiva optimizada, ambas como si no hubiera distorsión radial. El segundo caso se utiliza a veces para comparar las dos calibraciones optimizadas. En teoría, el error de reproyección al incluir la distorsión radial puede ser menor que el error de reproyección de la proyección lineal, sin embargo, en las buenas cámaras como las utilizadas para las pruebas experimentales casi no se nota, ya que la distorsión radial es despreciable. La inicialización de los parámetros de distorsión radial se comentará en la implementación.

6.7. Evaluación experimental

En este capítulo hay muchos algoritmos a evaluar. Iremos paso a paso aumentando el número de cámaras, de forma análoga a como se ha hecho en la introducción teórica de los algoritmos. Aplicaremos los conceptos generales explicados en § 4.6 sobre las medidas y los límites de los algoritmos de máxima verosimilitud.

6.7.1. Matriz fundamental

El algoritmo Gold Standard § 6.2.3.6 es el algoritmo de máxima verosimilitud y sus números son: $M = 3n + 12$ parámetros, $d = 3n + 7$ independientes y $N = 4n$, donde n es el número de correspondencias. Según las expresiones (4.3) y (4.4):

$$\begin{aligned} \epsilon_{\text{res}}^2 &= \sigma^2 \left(1 - \frac{d}{N}\right) &&= \sigma^2 \left(\frac{1}{4} - \frac{7}{4n}\right) \\ \epsilon_{\text{est}}^2 &= \sigma^2 \frac{d}{N} &&= \sigma^2 \left(\frac{3}{4} + \frac{7}{4n}\right) \end{aligned} \tag{6.32}$$

Los límites de los errores RMS de estas expresiones cuando el número de puntos tiende a infinito son $\sqrt{1/4}\sigma$ y $\sqrt{3/4}\sigma$, respectivamente. A medida que aumenta el número de puntos, disminuye la distancia de los puntos estimados $\hat{\mathbf{x}}_i$ a los exactos $\bar{\mathbf{x}}$ y aumenta la distancia de los puntos estimados $\hat{\mathbf{x}}_i$ a los puntos ruidosos $\mathbf{x}_i$, pero las ecuaciones no predicen una gran mejora. Las curvas que representan el error RMS normalizado por σ, $\epsilon_{\text{res}}/\sigma$ y $\epsilon_{\text{est}}/\sigma$, frente al número de puntos son las de la figura 6.12.

Descripción de los experimentos

Los datos sintéticos utilizados para la evaluación de los algoritmos de estimación de la matriz fundamental simulan la geometría epipolar: dos cámaras con una distancia focal de 20 mm, píxeles cuadrados y punto principal en el origen de coordenadas, apuntando a un conjunto de puntos uniformemente distribuidos dentro de una esfera de radio 1 m. Las posiciones de las cámaras (centros ópticos) son aleatorias sobre una superficie esférica, situadas a una distancia media de 8 m del centro de la esfera de puntos. Las cámaras están ligeramente desviadas, sin apuntar al centro de la esfera de puntos.

Los puntos exactos sobre las imágenes se obtienen proyectando los puntos 3D. Los puntos ruidosos se logran sumando ruido gaussiano de media cero y desviación típica σ conocida a los exactos.

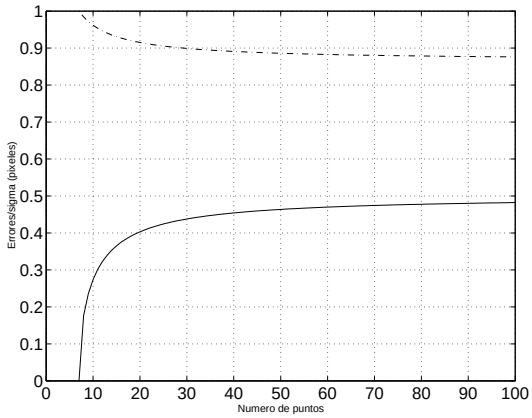


Figura 6.12: Matriz fundamental: límites teóricos de los errores RMS residual (línea continua) y de estimación (línea discontinua) del algoritmo Gold Standard (máxima verosimilitud)

Los algoritmos a evaluar son 3, principalmente: el algoritmo lineal normalizado de los 8 puntos (DLT−NA) que minimiza la distancia algebraica en el ambiente e impone la restricción de singularidad a posteriori, el algoritmo que minimiza la distancia algebraica (MAD − Min Algebraic Distance) dentro del conjunto de las matrices singulares y el algoritmo que minimiza el error de reproyección (GS − Gold Standard).

No hace falta evaluar el algoritmo de los 7 puntos porque siempre consigue una solución exacta y si le pasamos puntos ruidosos, será solución exacta respecto de esos datos, no respecto de los exactos. De los experimentos del GS se puede sacar un valor aproximado del umbral de clasificación de los inliers para el algoritmo que utiliza RANSAC.

Para cada algoritmo se evalúa su coste propio: cómo varían las mínimas distancias algebraicas *normalizadas* en los algoritmos algebraicos y cómo varía el error de reproyección en el Gold Standard.

En este caso, como en el de la homografía, tampoco es fácil elegir un parámetro de calidad sobre el cual comparar los tres algoritmos. Los dos algoritmos algebraicos sólo devuelven la matriz fundamental, así que debemos comparar las estimaciones de este objeto, sin puntos corregidos. El criterio de comparación es la distancia de un punto a su recta epipolar correspondiente, tanto para datos ruidosos como para datos exactos. Este criterio no sigue las curvas teóricas de la figura 6.12. Las ecuaciones de estos parámetros de calidad son:

$$\begin{aligned} PQ^2_{\text{res}} &= \frac{1}{n}\sum_{i=1}^{n}\left[d(\mathbf{x}_i,\mathbf{l}_i)^2 + d(\mathbf{x}'_i,\mathbf{l}'_i)^2\right] \quad \text{donde} \quad \mathbf{l}_i = \mathtt{F}^\top\mathbf{x}'_i, \mathbf{l}'_i = \mathtt{F}\mathbf{x}_i \\ PQ^2_{\text{est}} &= \frac{1}{n}\sum_{i=1}^{n}\left[d(\bar{\mathbf{x}}_i,\hat{\mathbf{l}}_i)^2 + d(\bar{\mathbf{x}}'_i,\hat{\mathbf{l}}'_i)^2\right] \quad \text{donde} \quad \hat{\mathbf{l}}_i = \mathtt{F}^\top\bar{\mathbf{x}}'_i, \hat{\mathbf{l}}'_i = \mathtt{F}\bar{\mathbf{x}}_i \end{aligned} \tag{6.33}$$

La distancia de un punto a una recta es la ecuación (6.9).

Los valores experimentales elegidos para la desviación típica de ruido σ son los mismos que para evaluar homografías: de 0 a 5 en pasos de 0.5. También mantenemos el número de experimentos realizados $n_{\text{exp}} = 200$. El número de puntos varía dentro de los permitidos ($n \geq 7$), que son $n = \{8, 10, 15, 20, 40, 70, 100\}$.

Costes propios

Veamos cómo se cumplen las ecuaciones teóricas para el algoritmo Gold Standard. Las figuras 6.13 y 6.14 muestran los resultados experimentales de los errores RMS residual $\boldsymbol{\epsilon}_{\text{res}}$ y de estimación $\boldsymbol{\epsilon}_{\text{est}}$ de dicho algoritmo, para $n = \{10, 20, 40, 100\}$ puntos.

En esta exposición no ponemos todas las gráficas para no abrumar al lector, sólo las más relevantes. Comparemos la evolución de estas gráficas con los valores esperados que indica la gráfica teórica (figura 6.12): se verifica que a medida que crece el número de puntos el error residual aumenta mientras que el error de estimación disminuye, además se observa que las medidas para un número elevado de puntos ($n = 100$) se acercan a los valores de los límites teóricos: $\boldsymbol{\epsilon}_{\text{res}}/\sigma = \sqrt{1/4} = 0{,}5$ y $\boldsymbol{\epsilon}_{\text{est}}/\sigma = \sqrt{3/4} \approx 0{,}866$ (ver la gráfica de la derecha de la figura 6.14).

Una vez más, las curvas teóricas se cumplen, no sólo cualitativamente, sino cuantitativamente.

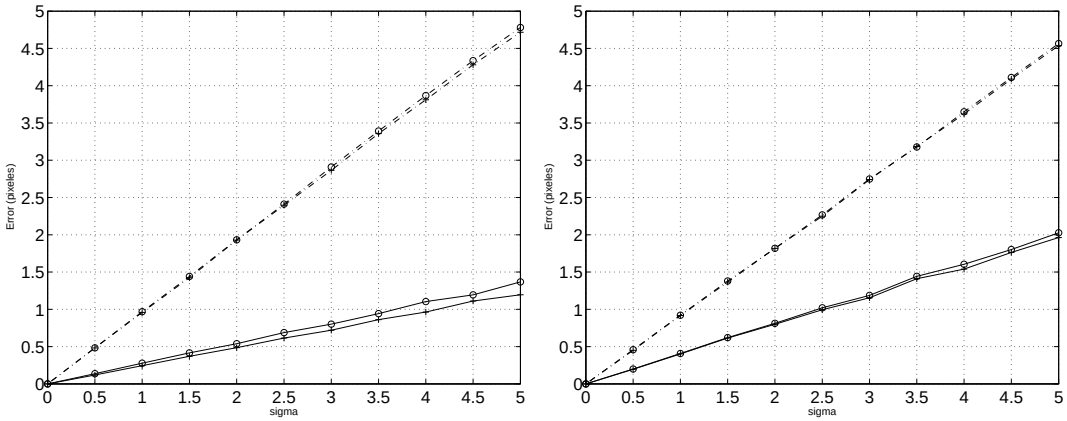


Figura 6.13: Matriz fundamental: errores RMS residual y de estimación de los algoritmos lineal y Gold Standard con $n = 10$ puntos (izquierda) y $n = 20$ (derecha).

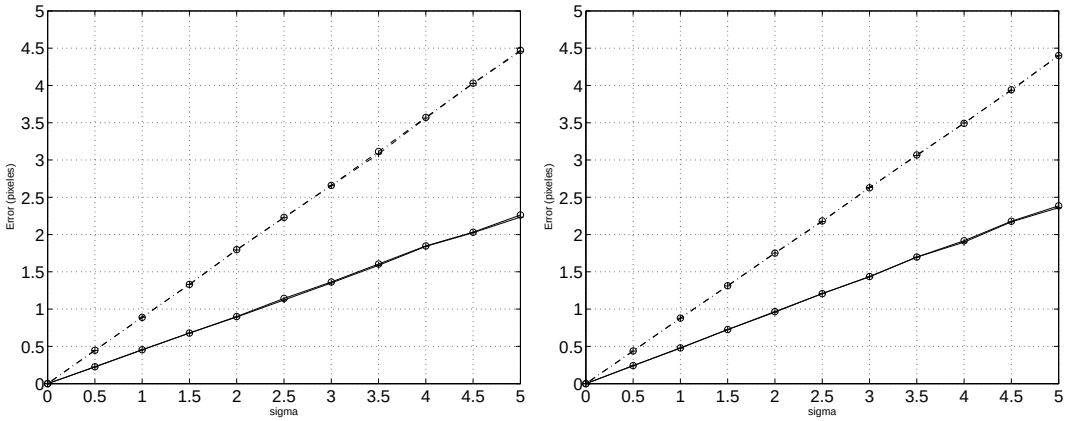


Figura 6.14: Matriz fundamental: errores RMS residual y de estimación de los algoritmos lineal y Gold Standard con $n = 20$ puntos (izquierda) y $n = 100$ (derecha)

En esta ocasión el error residual siempre se mantiene por debajo del error de estimación: no se produce el fenómeno de inversión de los errores. Como se aprecia en las gráficas, los valores medio y mediana de las estimaciones convergen si crece n.

Otro comentario sobre los costes propios se refiere a los algoritmos algebraicos: las gráficas de la distancia mínima algebraica normalizada frente a σ siguen una variación lineal y no se aprecia una diferencia significativa entre la distancia mínima alcanzada por el algoritmo DLT−NA y la alcanzada por el algoritmo MAD: ambas rectas se superponen.

Comparación respecto del parámetro de calidad

Las figuras 6.15 y 6.16 recogen los resultados experimentales de la variación de la distancia punto-recta epipolar frente al ruido, tanto residual $\boldsymbol{\epsilon}_{\text{res}}$ como de estimación $\boldsymbol{\epsilon}_{\text{est}}$ y para los valores del número de puntos seleccionado: $n = \{10, 20, 40, 100\}$. Seguimos la notación habitual de las gráficas, dada en experimentos anteriores.

No tenemos ninguna curva teórica precedente con la cual comparar los resultados de las pruebas realizadas con este parámetro de calidad, luego sólo podemos comentar en líneas generales la evolución de las gráficas con n y σ. Empecemos con la variación con σ: una vez más la respuesta de los algoritmos es lineal, sin embargo la escala vertical ha cambiado. Según las fórmulas (6.33), la normalización consiste en dividir por n, no por el número de medidas del algoritmo GS ($4n$). Se debe llevar cuidado con este pequeño detalle. Quizá lo correcto hubiera sido dividir por $2n$, ya que la distancia de un punto a una recta es una medida y si hay n puntos en 2 imágenes, tenemos $2n$ medidas. En cualquier caso, una medida es proporcional a otra. Otra puntualización: las gráficas, al igual que antes, están mejor estimadas ("son más lineales") cuantos más puntos se utilicen.

Para este parámetro de calidad, siempre se obtienen mejores resultados con el algoritmo GS que con el MAD, y a su vez, mejores estimaciones con el MAD que con el DLT−NA. La diferencia de estimaciones entre los algoritmos DLT−NA y MAD son menores cuantos más puntos se utilicen, llegando, por ejemplo, a ser indistinguibles para $n = 100$ puntos.

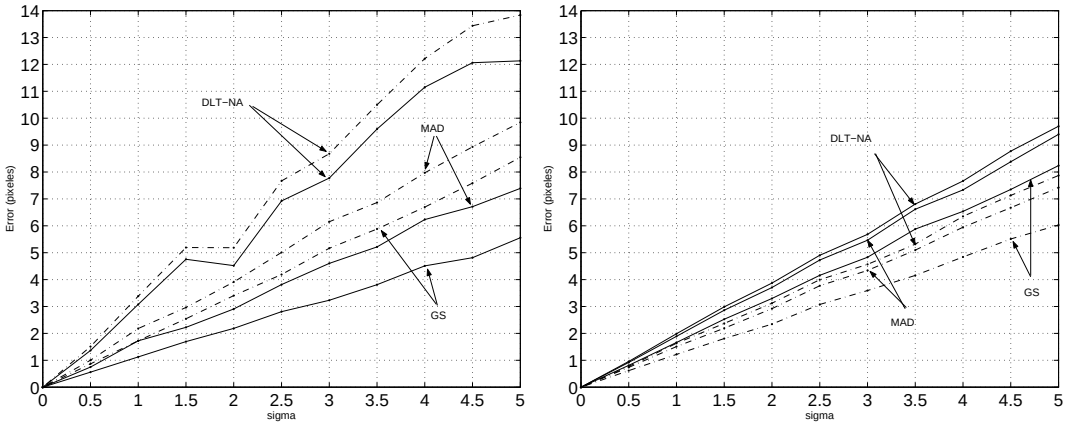


Figura 6.15: Matriz fundamental: valores RMS residual y de estimación del error distancia punto-recta epipolar con $n = 10$ puntos (izquierda) y $n = 20$ (derecha).

También se observa el fenómeno de inversión de los errores residual y de estimación: según crece el número de puntos, el error residual aumenta y el error de estimación disminuye. La inversión se produce para algún número de puntos entre $n = 10$ y $n = 20$, el cual no ha sido determinado.

6.7.2. Triangulación

Los datos sintéticos utilizados para la estimación de los puntos 3D son muy parecidos a los empleados para la matriz fundamental: se generan dos cámaras y unos puntos 3D uniformemente distribuidos

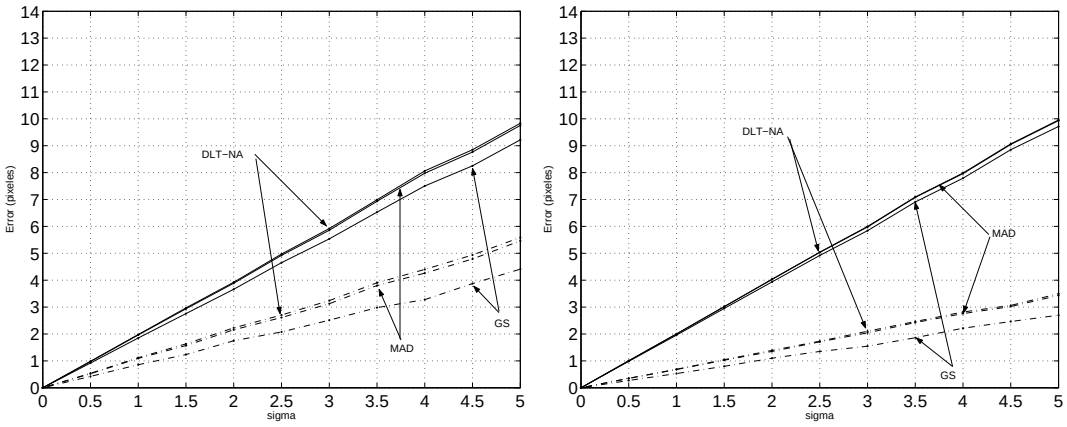


Figura 6.16: Matriz fundamental: valores RMS residual y de estimación del error distancia punto-recta epipolar con $n = 40$ puntos (izquierda) y $n = 100$ (derecha)

dentro de una esfera. A continuación se proyectan y se obtienen los puntos exactos. Por último se añade ruido gaussiano de media nula y varianza conocida a cada coordenada de los puntos, como es habitual.

Para probar los algoritmos de triangulación lineal (DLT) y óptimo (OPT) se utilizan los puntos ruidosos y las matrices de proyección exactas de las cámaras generadas en la simulación. De esta forma se mide cómo responde el algoritmo de triangulación al ruido en las imágenes, sin incluir ruido en las matrices de proyección.

El parámetro de calidad con el que se comparan ambos algoritmos es el error de reproyección de los puntos 3D estimados, tanto el error residual (respecto de los puntos ruidosos) como el error de estimación (respecto de los puntos exactos). Las ecuaciones de este parámetro, en sus dos versiones son:

$$\begin{aligned} PQ^2_{\text{res}} &= \frac{1}{4n}\sum_{i=1}^{n}\left[d(\mathbf{x}_i,\hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i,\hat{\mathbf{x}}'_i)^2\right] \\ PQ^2_{\text{est}} &= \frac{1}{4n}\sum_{i=1}^{n}\left[d(\bar{\mathbf{x}}_i,\hat{\mathbf{x}}_i)^2 + d(\bar{\mathbf{x}}'_i,\hat{\mathbf{x}}'_i)^2\right] \end{aligned} \quad , \tag{6.34}$$

donde $\hat{\mathbf{x}}_i = \bar{\mathtt{P}}\hat{\mathbf{X}}_i$ y $\hat{\mathbf{x}}'_i = \bar{\mathtt{P}}'\hat{\mathbf{X}}_i$. El término de normalización $1/4n$ tiene en cuenta que hay 2 puntos proyectados (4 coordenadas) asociados a un mismo punto 3D.

Ésta es, sin duda alguna, la simulación más rápida de cuantas se han presentado, ya que la triangulación se hace para cada punto por separado, así que es la misma independientemente del número de puntos que se utilice: no hay variación de las curvas con el número de puntos, sólo variación con la desviación típica de ruido. La figura 6.17 es el resultado de las pruebas.

Se observa que siempre se obtienen mejores resultados con el algoritmo OPT que con el algoritmo DLT, sin embargo, no hay mucha diferencia. Otra observación es, como cabía esperar, que el error de estimación es siempre mayor que el error residual, aproximadamente el doble. Esto sa debe a que hay muy pocos datos como para que la estimación se acerque más a los puntos exactos que a los puntos ruidosos.

Los valores aproximados de las 4 rectas de la figura son: para el algoritmo lineal, $\boldsymbol{\epsilon}_{\text{res}}/\sigma = 0{,}5106$ y $\boldsymbol{\epsilon}_{\text{est}}/\sigma = 0{,}876$; para el algoritmo óptimo, $\boldsymbol{\epsilon}_{\text{res}}/\sigma = 0{,}5029$ y $\boldsymbol{\epsilon}_{\text{est}}/\sigma = 0{,}863$.

Para unas pruebas mucho más detalladas se puede consultar [4].

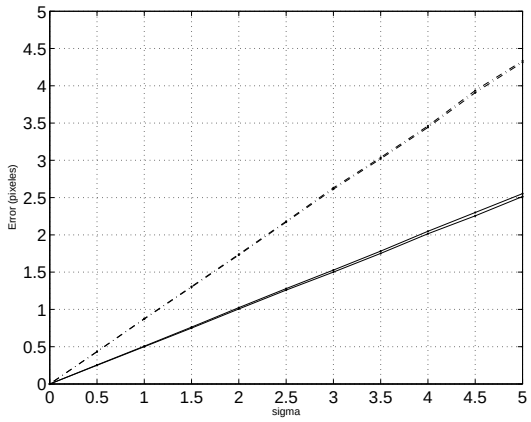


Figura 6.17: Triangulación: valores RMS residual y de estimación del error de reproyección debido a los algoritmos de triangulación.

6.7.3. Tensor trifocal

El algoritmo de máxima verosimilitud es el Gold Standard § 6.4.3.3 y sus números son: $M = 3n + 24$ parámetros, $d = 3n + 18$ independientes según la tabla 6.3 y $N = 6n$, donde n es el número de correspondencias. Según las expresiones (4.3) y (4.4):

$$\begin{aligned} \epsilon_{\text{res}}^2 &= \sigma^2\left(1 - \frac{d}{N}\right) = \sigma^2\left(\frac{1}{2} - \frac{3}{n}\right) \\ \epsilon_{\text{est}}^2 &= \sigma^2\frac{d}{N} \qquad\quad = \sigma^2\left(\frac{1}{2} + \frac{3}{n}\right) \end{aligned} \tag{6.35}$$

Los límites de los errores RMS de estas expresiones cuando el número de puntos tiende a infinito son $\sqrt{1/2}\sigma$ en ambos casos. Se sigue cumpliendo que conforme aumenta el número de puntos, disminuye la distancia de los puntos estimados $\hat{\mathbf{x}}_i$ a los exactos $\bar{\mathbf{x}}$ y aumenta la distancia de los puntos estimados $\hat{\mathbf{x}}_i$ a los puntos ruidosos $\mathbf{x}_i$, esta vez los límites son mejores que los de la matriz fundamental porque hay una cámara más (más restricciones). Las curvas que representan el error RMS normalizado por σ, $\epsilon_{\text{res}}/\sigma$ y $\epsilon_{\text{est}}/\sigma$, frente al número de puntos se recogen en la figura 6.18.

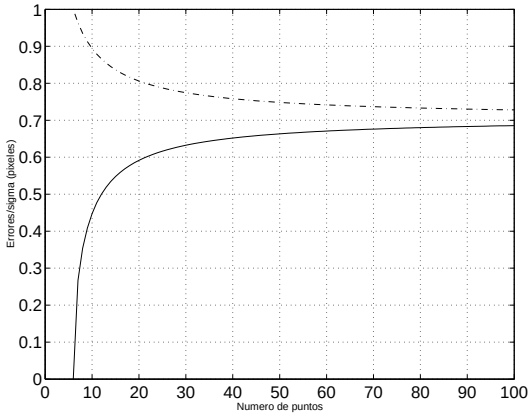


Figura 6.18: Tensor trifocal: límites teóricos de los errores RMS residual (línea continua) y de estimación (línea discontinua) del algoritmo Gold Standard (máxima verosimilitud)

Descripción de los experimentos

Los datos sintéticos utilizados para la evaluación de los algoritmos de estimación del tensor trifocal son análogos a los que se generan para estimar la matriz fundamental, pero con tres cámaras en lugar de dos.

En principio se decidió evaluar el algoritmo lineal con normalización afín, pero dado que no proporciona un tensor trifocal geométricamente válido ya que no se imponen las restricciones propias, algunos resultados salen disparatados. Así pues, en lugar de este algoritmo se evalúa el resultado de la primera iteración del algoritmo de minimización de la distancia algebraica, que sí proporciona un tensor trifocal con todas las restricciones propias. Además de esta primera iteración del algoritmo (MAD1), se evalúan el algoritmo que minimiza la distancia algebraica (MAD) y el algoritmo que minimiza el error de reproyección (GS − Gold Standard).

Tampoco en esta ocasión es necesario evaluar el algoritmo de los 6 puntos porque siempre consigue una solución exacta y si le damos puntos ruidosos, será solución exacta respecto de esos puntos, no respecto de los exactos.

Para cada algoritmo se evalúa su coste propio: cómo varían las mínimas distancias algebraicas *normalizadas* en los algoritmos algebraicos y cómo varía el error de reproyección en el Gold Standard.

La elección del parámetro de calidad sobre el cual comparar todos los algoritmos es más complicada que las anteriores. Los dos algoritmos algebraicos sólo devuelven el tensor trifocal; una vez más se desconocen los puntos corregidos.

El criterio de comparación es una variante del error de reproyección: a partir del tensor trifocal se extraen las matrices de proyección, según se explica en § 6.4.4; luego se triangula para estimar las posiciones de los puntos 3D que dan lugar a las observaciones; por último, se proyectan los puntos 3D estimados y se mide su distancia respecto de los datos ruidosos y respecto de los datos exactos. El algoritmo GS es el único que no necesita triangulación porque la integra internamente, pero hay que hacerla de igual forma para tratarlo igual que los demás algoritmos.

Este criterio no sigue las curvas teóricas de la figura 6.18, ya que en él influye el algoritmo de triangulación. Las ecuaciones de los parámetros de calidad son:

$$\begin{aligned} PQ^2_{\text{res}} &= \frac{1}{6n}\sum_{i=1}^{n}\left[d(\mathbf{x}_i,\hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i,\hat{\mathbf{x}}'_i)^2 + d(\mathbf{x}''_i,\hat{\mathbf{x}}''_i)^2\right] \\ PQ^2_{\text{est}} &= \frac{1}{6n}\sum_{i=1}^{n}\left[d(\bar{\mathbf{x}}_i,\hat{\mathbf{x}}_i)^2 + d(\bar{\mathbf{x}}'_i,\hat{\mathbf{x}}'_i)^2 + d(\bar{\mathbf{x}}''_i,\hat{\mathbf{x}}''_i)^2\right] \end{aligned} \quad . \tag{6.36}$$

El término $1/6n$ normaliza el coste para cada coordenada del vector de medidas: si hay n correspondencias de puntos en 3 imágenes, hay $6n$ coordenadas.

Los valores experimentales utilizados son: σ de 0 a 5 en pasos de 0.5 y 100 experimentos, ya que cada experimento es más costoso que el equivalente para la matriz fundamental. El número de puntos que se han elegido en las simulaciones son: $n = \{7, 8, 10, 20, 40, 70, 100\}$. El valor $n = 7$ es el primer valor que podemos tomar para el que hay solución lineal por mínimos cuadrados, sin embargo los resultados son mejores cuantos más puntos se utilicen.

Costes propios

Anteriormente no se han presentado en detalle las gráficas de la variación de la mínima distancia algebraica frente al ruido y el número de puntos. En esta ocasión sí incluimos las gráficas (figuras 6.19

y 6.20), que recogen la evolución de las distancias algebraicas normalizadas de los algoritmos MAD1 y MAD.

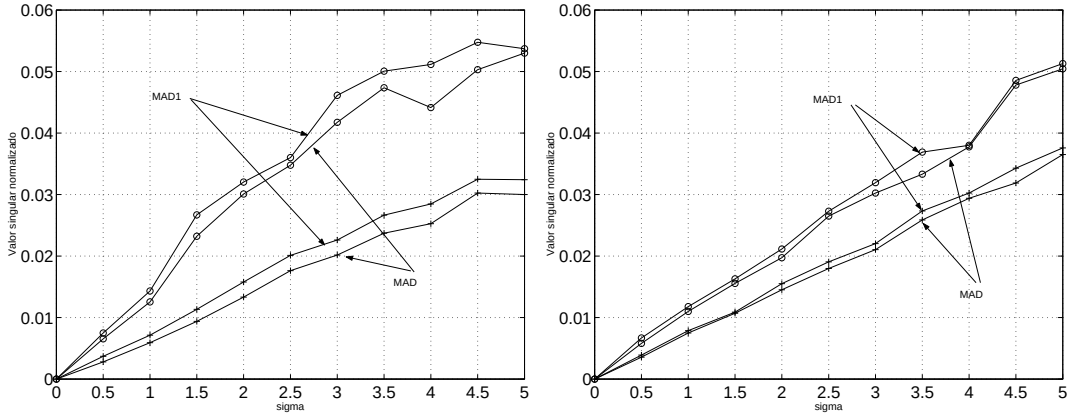


Figura 6.19: Tensor trifocal: distancias algebraicas normalizadas de los algoritmos MAD1 y MAD para $n = 8$ puntos (izquierda) y $n = 10$ (derecha).

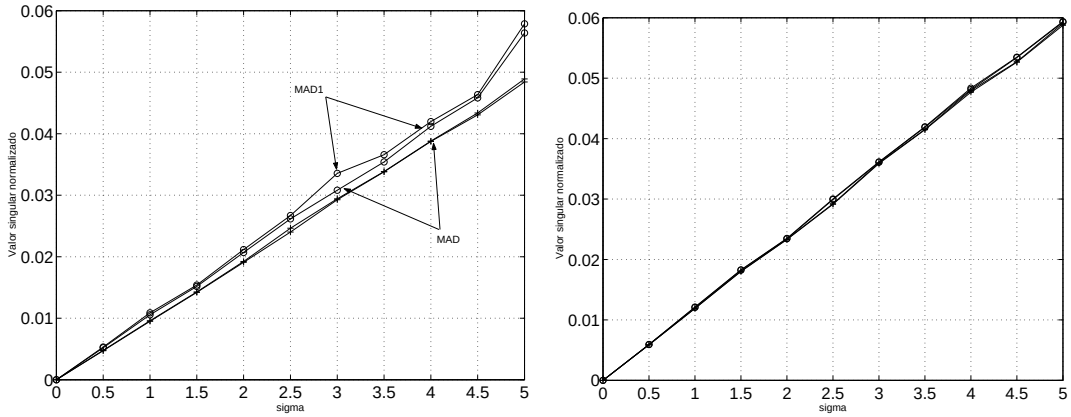


Figura 6.20: Tensor trifocal: distancias algebraicas normalizadas de los algoritmos MAD1 y MAD para $n = 20$ puntos (izquierda) y $n = 100$ (derecha)

Comparándolas entre sí se observa que, en general, las mínimas distancias algebraicas siguen una variación lineal con la desviación típica de ruido. Con $n = 8$ puntos no parece claro que los valores medios sigan dicha variación, sin embargo, a medida que crece el número de puntos, las rectas sobre las que se encuentran dichos valores medios están mejor definidas, poseen menor incertidumbre, llegando a ser prácticamente coincidentes para $n = 100$ puntos. También se consigue que los estimadores media y mediana confluyan.

Asimismo, se han obtenido las curvas de variación del coste del algoritmo GS (error residual), junto con su error de estimación correspondiente. Las figuras 6.21 y 6.22 son la prueba de ello. La información contenida en estas curvas es la que hay que comparar con la predicción de las curvas teóricas (figura 6.18).

Se aprecia una variación lineal del error de reproyección en función de la desviación típica de ruido. Quizá con $n = 8$ los resultados se ajustan peor a una recta que con $n = 100$, mas ambos siguen

teóricamente la misma variación lineal con σ. A medida que crece n, las estimaciones son mejores, lo que se aprecia en la confluencia de los valores medio y mediana, además de que cada vez son más rectas las curvas. Como predicen las ecuaciones teóricas, siempre es mayor el error de estimación que el error residual y según crece n esta diferencia disminuye, tendiendo a anularse: por ejemplo, para $n = 100$ y $\sigma = 5$ los valores experimentales son $\boldsymbol{\epsilon}_{\text{est}}/\sigma = 0{,}7383$ y $\boldsymbol{\epsilon}_{\text{res}}/\sigma = 0{,}6915$, que se ajustan bien a los valores teóricos: 0,7280 y 0,6856, respectivamente, todos cercanos al límite $1/\sqrt{2} \approx 0{,}7071$. Comprobamos que, según lo que dicta la teoría, no se produce el fenómeno de inversión de las curvas.

Las curvas obtenidas siguen cuantitativamente los valores esperados de las curvas teóricas, lo que justifica que los algoritmos programados son eficaces, tanto como el óptimo.

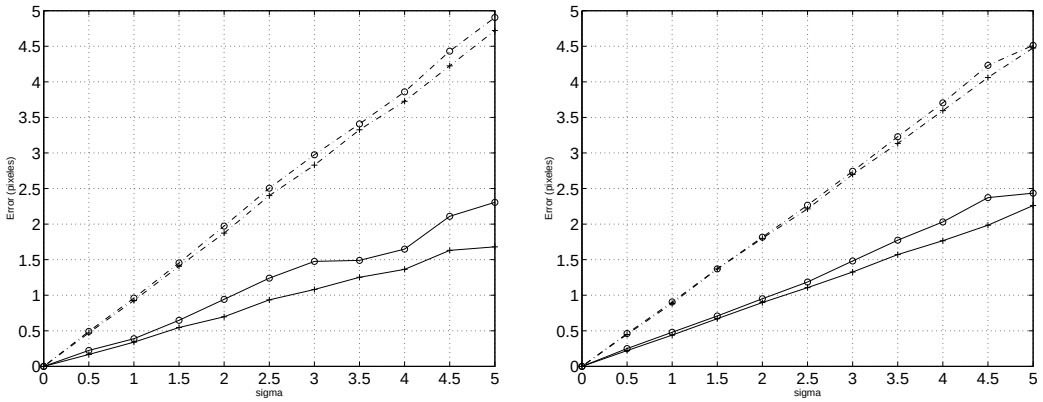


Figura 6.21: Tensor trifocal: valores RMS residual y de estimación del error de reproyección con $n = 8$ puntos (izquierda) y $n = 10$ (derecha).

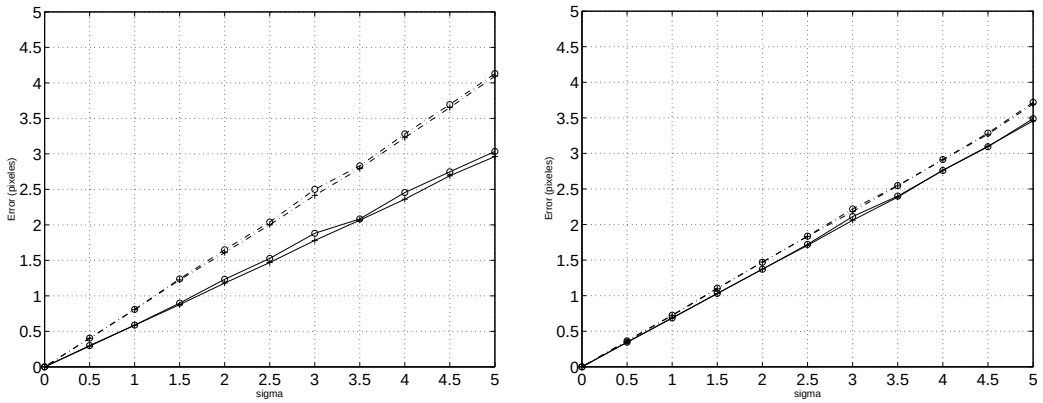


Figura 6.22: Tensor trifocal: valores RMS residual y de estimación del error de reproyección con $n = 20$ puntos (izquierda) y $n = 100$ (derecha)

Comparación respecto del parámetro de calidad

Las figuras 6.23 y 6.24 muestran los resultados experimentales de la variación del parámetro de calidad (6.36), elegido para comparar los tres algoritmos entre sí. Recogen tanto la variación frente al ruido (residual PQ_{res} y de estimación PQ_{est}) como la variación frente al número de puntos utilizados: $n = \{8, 10, 20, 100\}$.

La respuesta de los algoritmos es aproximadamente lineal con la desviación típica de ruido. La normalización empleada para los datos es la misma que la marcada por el error RMS de reproyección del algoritmo GS ($1/6n$), sin embargo los errores debidos a este parámetro de calidad son un poco mayores que los de las últimas pruebas.

A pesar de que en el parámetro de calidad influye la triangulación, siempre se obtienen mejores resultados con el algoritmo GS que con el MAD, y a su vez, mejores estimaciones con el MAD que con la primera iteración (MAD1). La diferencia de resultados entre los algoritmos MAD1 y MAD son menores cuantos más puntos se utilicen, llegando, por ejemplo, a ser prácticamente coincidentes para $n = 100$ puntos.

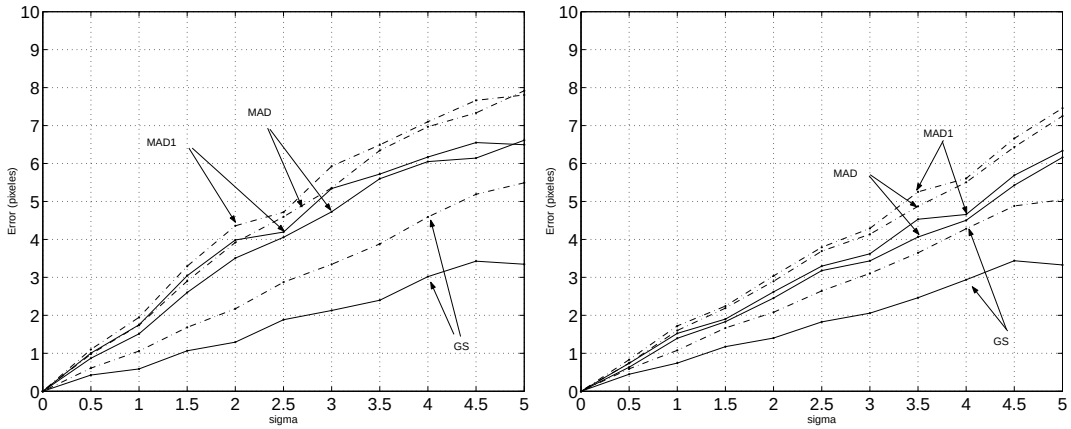


Figura 6.23: Tensor trifocal: valores RMS residual y de estimación del error de reproyección aproximado, para $n = 8$ puntos (izquierda) y $n = 10$ (derecha).

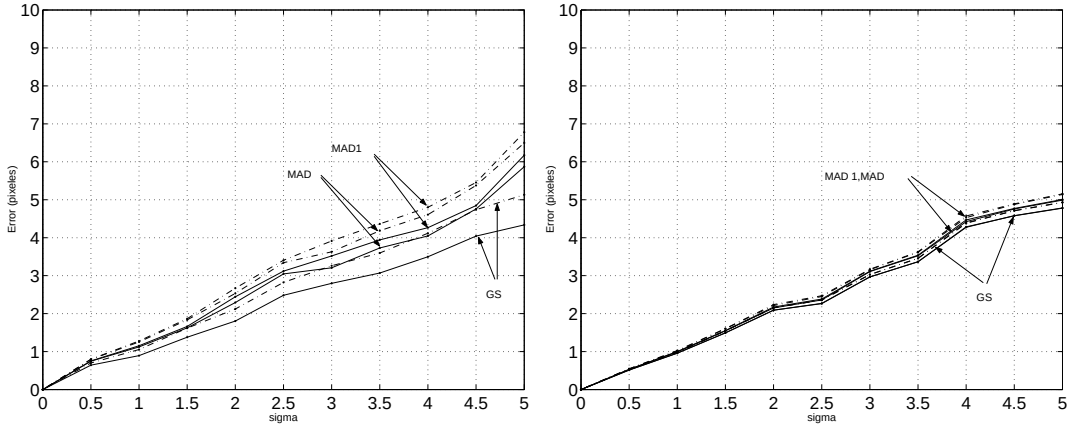


Figura 6.24: Tensor trifocal: valores RMS residual y de estimación del error de reproyección aproximado, para $n = 20$ puntos (izquierda) y $n = 100$ (derecha)

Con este parámetro de calidad tampoco se observa el fenómeno de inversión de los errores. Las estimaciones siguen las tendencias de las curvas teóricas en líneas generales, pero con distintos valores para el error.

En general, podemos concluir que no es recomendable estimar el tensor trifocal sólo con $n = 8$ puntos, sino que merece la pena utilizar unos pocos más, por ejemplo $n = 20$, ya que los resultados son mejores. En realidad, si se utiliza el algoritmo GS, uno debe mirar las curvas teóricas (figura 6.18) y decidir dónde quiere trabajar, ya que los resultados obtenidos en los experimentos siguen esas curvas.

6.7.4. Tensor cuadrifocal

Los límites teóricos de los mínimos errores obtenibles utilizando el algoritmo de máxima verosimilitud (implementado de forma general en el ajuste de haces proyectivo) para 4 cámaras se obtienen con los siguientes números: $d = 3n + 29$, $3n$ para las coordenadas (afines) de los puntos en el espacio, más 29 para los grados de libertad del tensor cuadrifocal y el número de medidas, $N = 8n$, porque hay n puntos (2 coordenadas) en cada imagen.

$$\begin{aligned} \epsilon_{\text{res}}^2 &= \sigma^2\left(1 - \frac{d}{N}\right) &&= \sigma^2\left(\frac{5}{8} - \frac{29}{8n}\right) \\ \epsilon_{\text{est}}^2 &= \sigma^2\frac{d}{N} &&= \sigma^2\left(\frac{3}{8} + \frac{29}{8n}\right) \end{aligned} \tag{6.37}$$

El límite del error residual RMS cuando $n \to \infty$ es $\sqrt{5/8}\sigma$ y el límite del error de estimación RMS es $\sqrt{3/8}\sigma$. Por primera vez en la geometría de varias cámaras la distancia de los puntos estimados respecto de los puntos exactos es menor que la distancia respecto de los puntos ruidosos. Las curvas que representan el error RMS normalizado por σ, $\epsilon_{\text{res}}/\sigma$ y $\epsilon_{\text{est}}/\sigma$, frente al número de puntos son las de la figura 6.25. Es apreciable que los dos errores son aproximadamente iguales para $n = 30$ puntos.

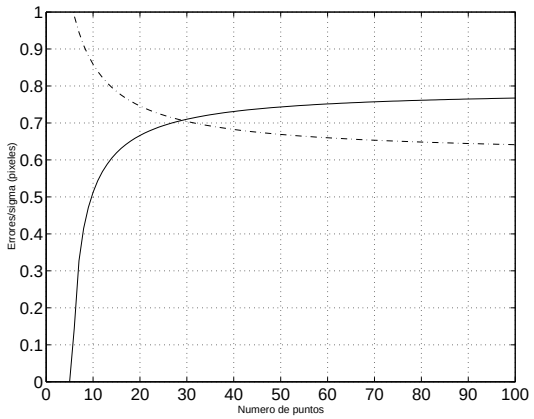


Figura 6.25: Tensor cuadrifocal: límites teóricos de los errores RMS residual (línea continua) y de estimación (línea discontinua) del algoritmo Gold Standard para el tensor cuadrifocal (máxima verosimilitud)

Descripción de los experimentos

Los datos sintéticos utilizados para la evaluación de los algoritmos de estimación del tensor cuadrifocal son análogos a los que se generan para estimar la matriz fundamental, pero con cuatro cámaras en lugar de dos.

En esta ocasión evaluamos el algoritmo de Heyden (HEY), como caso excepcional. Para el algoritmo lineal sencillo sucede lo mismo que para el tensor trifocal: como no se imponen las restricciones propias del tensor no siempre es posible obtener una configuración de cámaras que se ajuste al tensor estimado, es decir, que no es un tensor geométricamente válido y los resultados de las simulaciones

son disparatados. Se sigue la misma estrategia: se evalúa el resultado de la primera iteración del algoritmo de minimización de la distancia algebraica con restricciones propias. Además de esta primera iteración del algoritmo (MAD1), se evalúan los algoritmos que minimizan la distancia algebraica con las parametrizaciones en $\mathbb{R}^9$ (MAD$-$9) y $\mathbb{R}^{27}$ (MAD$-$27), junto con el algoritmo que minimiza el error de reproyección (GS $-$ Gold Standard).

Para cada algoritmo se evalúa su coste propio: cómo varían las mínimas distancias algebraicas *normalizadas* en los algoritmos algebraicos y cómo varía el error de reproyección en el Gold Standard.

Las mismas consideraciones que en la evaluación del tensor trifocal se pueden hacer sobre la elección del parámetro de calidad como criterio de comparación. Los algoritmos algebraicos sólo devuelven el tensor (o las matrices de proyección, de forma equivalente), no los puntos estimados. Se utiliza como criterio el error de reproyección en el que influye el algoritmo de triangulación.

$$\begin{aligned} PQ^2_{\text{res}} &= \frac{1}{8n}\sum_{i=1}^{n}\left[d(\mathbf{x}_i,\hat{\mathbf{x}}_i)^2 + d(\mathbf{x}'_i,\hat{\mathbf{x}}'_i)^2 + d(\mathbf{x}''_i,\hat{\mathbf{x}}''_i)^2 + d(\mathbf{x}'''_i,\hat{\mathbf{x}}'''_i)^2\right] \\ PQ^2_{\text{est}} &= \frac{1}{8n}\sum_{i=1}^{n}\left[d(\bar{\mathbf{x}}_i,\hat{\mathbf{x}}_i)^2 + d(\bar{\mathbf{x}}'_i,\hat{\mathbf{x}}'_i)^2 + d(\bar{\mathbf{x}}''_i,\hat{\mathbf{x}}''_i)^2 + d(\bar{\mathbf{x}}'''_i,\hat{\mathbf{x}}'''_i)^2\right] \end{aligned} \tag{6.38}$$

El término $1/8n$ normaliza el coste para cada coordenada del vector de medidas: n correspondencias de puntos en 4 imágenes hacen un total de $8n$ coordenadas observadas.

Para las pruebas realizadas se han elegido valores de σ entre 0 y 5 en pasos de 0.5. Debido a que las ejecuciones son cada vez más costosas, se han realizado menos experimentos, 50, aunque son suficientes para obtener buenas medias del error. El número de puntos que se han elegido en las simulaciones son: $n = \{8, 10, 20, 30, 100\}$. Como se verá más adelante, el valor $n = 30$ es muy interesante.

Un detalle de las pruebas: los algoritmos algebraicos necesitan identificar 3 puntos que no estén alineados en ninguna proyección para construir una homografía de $\mathbb{P}^2$ que permita utilizar el tensor cuadrifocal reducido. El criterio de no alineados se realiza, en todas las imágenes, tomando puntos de 3 en 3, hallando las 3 rectas que los unen 2 a 2 y calculando las distancias de cada recta al punto por el que no pasa. Las 3 distancias halladas se comparan con un umbral adaptativo, dependiente del rango máximo de las coordenadas de los puntos utilizados (ya que pueden o no haber sufrido una normalización afín previa) y si superan dicho umbral, se valida la terna de puntos como linealmente independientes. Todos los algoritmos algebraicos utilizan los mismos 3 puntos para pasar al mundo del tensor reducido.

Costes propios

Hay cinco algoritmos algebraicos a comparar según sus respectivos costes: distancias algebraicas normalizadas. La figura 6.26 muestra la variación de dichos costes en función de la desviación típica de ruido, sólo para dos números de puntos distintos: $n = 30$ y $n = 100$. Si se utilizan menos puntos, las distancias algebraicas son más irregulares, aunque a grandes rasgos siguen las tendencias que marcan estas dos gráficas.

Según se apreciará al comparar el error de reproyección, las distancias algebraicas normalizadas de los algoritmos MAD guardan relación con las distancias geométricas, en cambio, el coste algebraico del algoritmo de Heyden va por libre: que sea menor que los valores singulares de los algoritmos MAD no quiere decir que el error de reproyección sea menor. Esto es debido a que la matriz de diseño del algoritmo de Heyden se construye de forma distinta a la matriz de diseño de los algoritmos MAD, por lo que sus menores valores singulares no son comparables, a pesar de que se hallan incluido en la misma gráfica.

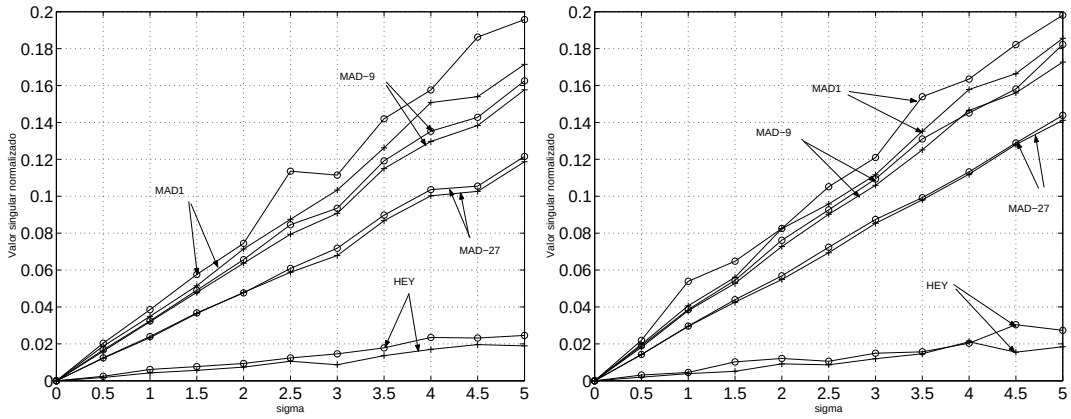


Figura 6.26: Tensor cuadrifocal: distancias algebraicas normalizadas de los algoritmos de estimación, para $n = 30$ puntos (izquierda) y $n = 100$ (derecha)

En concreto, para los algoritmos MAD se observa que la mayor distancia algebraica es la de la primera iteración (como era de esperar) y que la distancia del algoritmo MAD−27 es la menor de todas, por debajo de la del algoritmo MAD−9. Esto se debe a que la parametrización en $\mathbb{R}^{27}$ tiene más grados de libertad que la de $\mathbb{R}^9$ y permite hallar la solución en un espacio más amplio, de tal forma que se ajuste mejor, luego su distancia algebraica sea menor.

En general, las mínimas distancias algebraicas siguen una variación lineal con la desviación típica de ruido. Cuantos más puntos se utilicen, más rectas son las curvas.

Las figuras 6.27 y 6.28 son curvas experimentales de la variación del coste del algoritmo GS ($\boldsymbol{\epsilon}_{\text{res}}$) y el correspondiente error de estimación, $\boldsymbol{\epsilon}_{\text{est}}$. La información de estas curvas se debe contrastar con la que predicen las curvas teóricas (figura 6.25).

El error de reproyección varía de forma lineal con la desviación típica de ruido. Según aumenta n, vemos que se invierten las curvas de los errores residual y de estimación: como predicen las ecuaciones, no siempre es mayor el error de estimación que el error residual. A partir de $n = 30$ puntos, aproximadamente, la estimaciones están más cerca de los datos exactos que de los datos ruidosos.

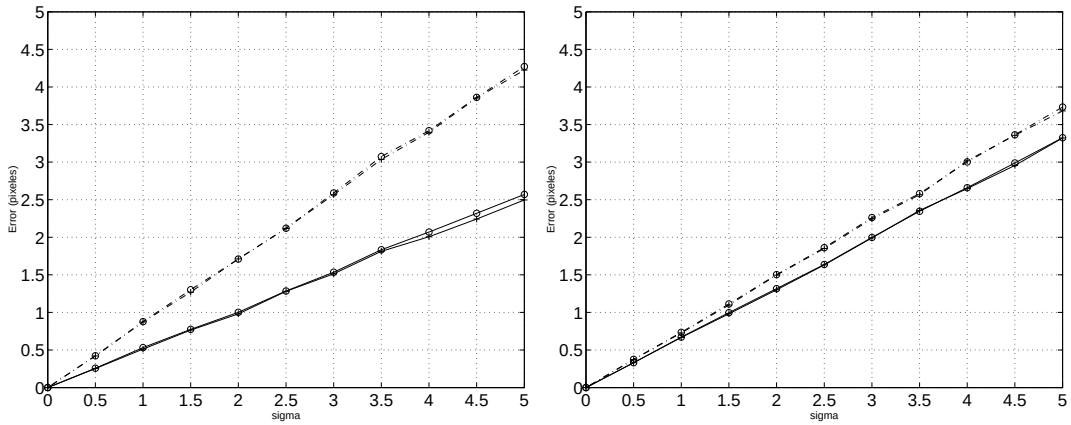


Figura 6.27: Tensor cuadrifocal: valores RMS residual y de estimación del error de reproyección con $n = 10$ puntos (izquierda) y $n = 20$ (derecha).

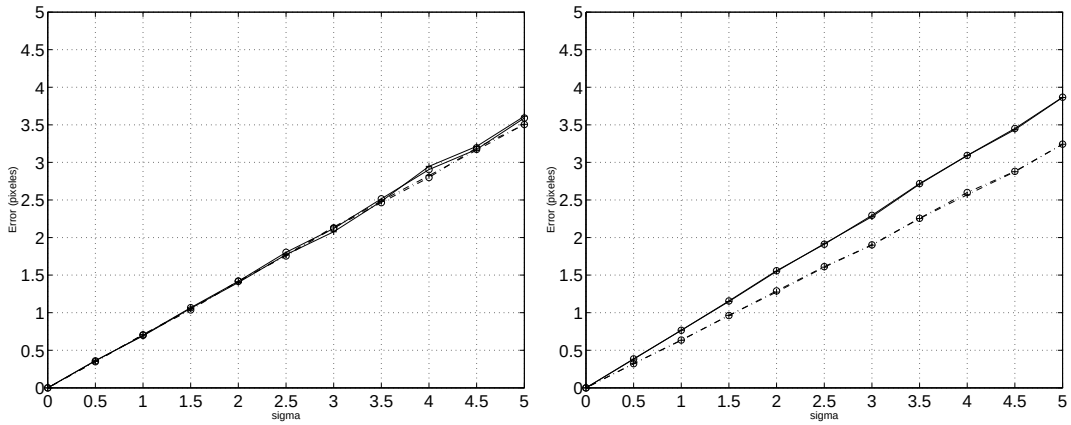


Figura 6.28: Tensor cuadrifocal: valores RMS residual y de estimación del error de reproyección con $n = 30$ puntos (izquierda) y $n = 100$ (derecha)

Es interesante ver por qué se han elegido estas gráficas (estos valores de n): para $n = 10$ el error residual es considerablemente menor que el error de estimación; para $n = 20$ la diferencia es menor y es muy parecida a la situación para $n = 100$, si se intercambian los papeles del error residual y el de estimación. Por último, $n = 30$ muestra el punto donde se cruzan los errores: se cumple la predicción de las curvas teóricas (figura 6.25), que afirma que en este valor son aproximadamente iguales el error residual y el error de estimación.

Comparación respecto del parámetro de calidad

Las figuras 6.29 y 6.30 muestran los resultados experimentales de la variación del parámetro de calidad (6.38), elegido para comparar todos los algoritmos entre sí. Recogen tanto la variación frente al ruido (residual PQ_{res} y de estimación PQ_{est}) como la variación frente al número de puntos utilizados: $n = \{10, 20, 30, 100\}$.

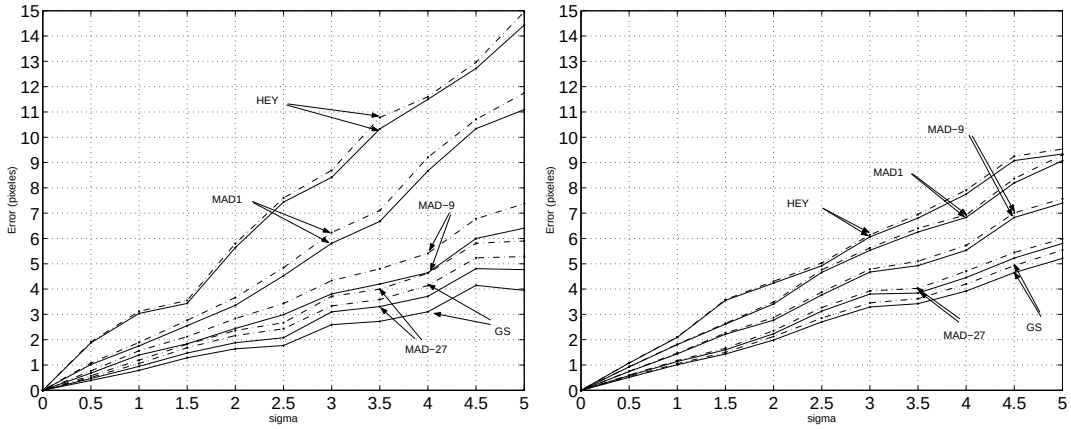


Figura 6.29: Tensor cuadrifocal: valores RMS residual y de estimación del error de reproyección aproximado, para $n = 10$ puntos (izquierda) y $n = 20$ (derecha).

Los algoritmos responden de forma lineal a la desviación típica de ruido, aproximadamente. La normalización empleada para los datos es la misma que la marcada por el error RMS de reproyección del algoritmo GS ($1/8n$), pero los errores debidos a este parámetro de calidad son mayores que los de

aquel, ya que en el parámetro de calidad influye la triangulación.

A pesar de todo, se cumple una jerarquía en los algoritmos, la que cabía esperar según su complejidad: los resultados del algoritmo GS son los mejores, seguidos de cerca por los del algoritmo MAD−27, a su vez mejores que los del algoritmo MAD−9 y que los algoritmos MAD1 y HEY, en este orden. La diferencia de estimaciones entre los algoritmos GS y MAD−27 decrece cuantos más puntos se utilicen. También se parecen cada vez más las estimaciones del algoritmo HEY a las del algoritmo MAD1.

La escala vertical de las gráficas de la figura 6.30 no es la misma que la escala de la figura 6.29; en unas, el eje vertical representa los errores (en píxeles) entre 0 y 15, mientras que en otras el intervalo es $[0, 10]$, para ver con mayor detalle las diferencias entre los algoritmos.

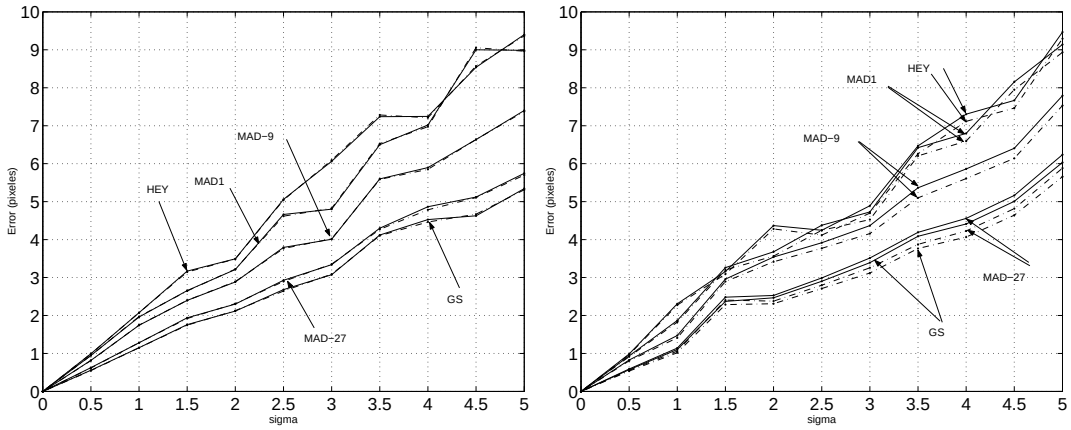


Figura 6.30: Tensor cuadrifocal: valores RMS residual y de estimación del error de reproyección aproximado, para $n = 30$ puntos (izquierda) y $n = 100$ (derecha)

Este parámetro de calidad también acusa la inversión de los errores predicha para el Gold Standard, sin embargo no es tan grande como la de aquel. Si se utiliza el algoritmo GS, es recomendable apreciar las curvas teóricas (figura 6.25) y elegir el punto de trabajo, pues los resultados experimentales coinciden con los teóricos.

6.7.5. Geometría multicámara

Podemos seguir aumentando el número de cámaras y sacar las expresiones de los errores residual y de estimación mínimos alcanzables por los algoritmos Gold Standard. En el caso de varias cámaras este algoritmo es el que realiza el ajuste de haces § 6.6.4. Como ya se ha resumido en § 6.6.1, para una calibración proyectiva de m cámaras y n puntos, el número de grados de libertad del sistema es $d = 3n + 11m - 15$ y el número de medidas es $N = 2mn$. Las expresiones de los límites de los errores en este caso general son, de acuerdo a las expresiones (4.3) y (4.4):

$$\begin{aligned} \epsilon^2_{\text{res}} &= \sigma^2\left(1 - \frac{d}{N}\right) = \sigma^2\left(1 - \frac{3n + 11m - 15}{2mn}\right) \\ \epsilon^2_{\text{est}} &= \sigma^2\frac{d}{N} \qquad\quad = \sigma^2\left(\frac{3n + 11m - 15}{2mn}\right) \end{aligned} \tag{6.39}$$

Sin tener en cuenta la variación con la desviación típica de ruido (lineal), podemos representar las superficies generadas por estas ecuaciones, parametrizadas en función de m y n, suponiendo que variasen en el campo de los números reales. La figura 6.31 recoge dichas superficies y las compara con las

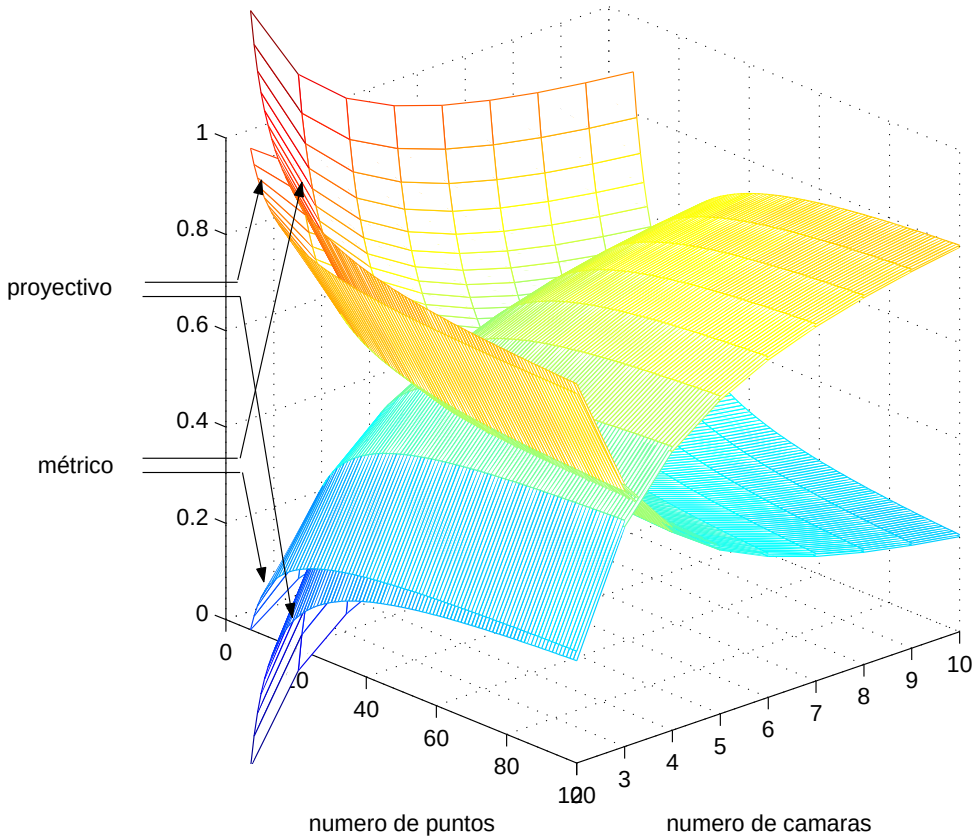


Figura 6.31: Geometría multicámara: límites teóricos de los errores residual y de estimación de los algoritmos de máxima verosimilitud, tanto para el ajuste de haces proyectivo como para el ajuste de haces métrico

de una calibración métrica, cuyos grados de libertad han cambiado: $d = 3n + 11m - 7$.

El eje vertical tiene las unidades habituales de Error dividido por la desviación típica de ruido y su rango de valores es el intervalo $[0, 1]$, aunque algunas curvas se salgan para ciertos valores del número de puntos (no tienen sentido).

Las figuras 6.12, 6.18 y 6.25 son rodajas de las dos superficies proyectivas. Ahora se aprecia con mayor claridad el fenómeno de la inversión de las curvas de error de estimación y residual. Al principio, para $m = 2$ cámaras, el error de estimación siempre está por encima del error residual, para cualquier valor de n. Para $m = 3$ cámaras ya no hay tanta diferencia, aunque sigue estando por encima el error de estimación; ambos errores son asintóticamente convergentes. Para $m = 4$ cámaras, ya se cruzan los errores, aproximadamente en $n = 30$ puntos. Y a medida que aumenta el número de cámaras, cada vez se cruzan antes, con un menor valor del número de puntos n. En el límite, si $n, m \to \infty$, sería posible hacer que los puntos estimados coincidieran con los puntos exactos (distancia nula).

Comparando las ecuaciones de los límites de la calibración proyectiva y la calibración métrica, vemos que lo único que cambian es el término independiente del número de grados de libertad. Las superficies se diferencian para valores pequeños de m y n, pero conforme estos valores crecen las diferencias disminuyen: son asintóticamente iguales.

Curiosidad. ¿Cuál es la intersección de las superficies $\boldsymbol{\epsilon}^2_{\text{res}}/\sigma^2$ y $\boldsymbol{\epsilon}^2_{\text{est}}/\sigma^2$ de la figura 6.31? Curiosamente la respuesta a esta pregunta para los casos tratados es *una hipérbola*.

Los puntos de intersección se hallan igualando las expresiones:

$$\frac{\boldsymbol{\epsilon}^2_{\text{res}}}{\sigma^2} = \frac{\boldsymbol{\epsilon}^2_{\text{est}}}{\sigma^2} \Leftrightarrow 1 - \frac{d}{N} = \frac{d}{N} \Leftrightarrow N = 2d \Leftrightarrow mn = 3n + 11m - 15$$

Así que la ecuación de la intersección es $\varphi \equiv nm - 3n - 11m + 15 = 0$. Ésta es una ecuación cuadrática en dos variables, o lo que es lo mismo: representa una cónica. La matriz de la cónica es

$$\mathtt{C}_\varphi \sim \begin{pmatrix} 0 & 1 & -3 \\ 1 & 0 & -11 \\ -3 & -11 & 30 \end{pmatrix}.$$

El determinante de la matriz de la cónica es no nulo, por lo que no es una cónica degenerada. Si hallamos la intersección de la ecuación de la cónica con la recta del infinito del plano (n, m), vemos que es $\mathbf{l}_\infty \cap \varphi \equiv nm = 0$. Lo que da dos puntos de corte: $n = 0$ y $m = 0$, que son las direcciones de los ejes coordenados m y n, respectivamente (puntos del infinito). Como hay dos puntos del infinito de corte distintos y la cónica es no degenerada, la cónica afín que representa es una hipérbola.

Descripción de los experimentos

Los datos sintéticos utilizados para la evaluación de los algoritmos de estimación de la calibración proyectiva multicámara son similares a los generados para estimar la matriz fundamental, pero con más de cuatro cámaras. En concreto, en los experimentos se han simulado $m = 10$ cámaras, por ser las mínimas que necesita el algoritmo de autocalibración basado en el Calibration Pencil.

Para la inicialización de la calibración proyectiva se han realizado pruebas con dos de los representantes de las dos familias introducidas en § 6.6.3, los que utilizan la matriz fundamental. Con pocas pruebas preliminares se concluye que los métodos que no utilizan resección (los que realizan cambios de referencia en el espacio) son ineficaces, ya que dan resultados disparatados. Esto se debe a la imposibilidad de elegir una verdadera referencia espacial y numéricamente estable, pues no hay métrica en $\mathbb{P}^3$ que induzca un criterio de comparación.

Además, los resultados varían significativamente en función de los puntos que se elijan para realizar el cambio de referencia y esto es una situación desagradable. Durante todo el proyecto se pretende elegir algoritmos que resuelvan problemas independientemente del orden en que sean aportados los datos, de tal forma que traten todos esos datos por igual, sin que ninguno prevalezca sobre otros.

Así que en esta ocasión se evalúa sólo el algoritmo de inicialización (INIC) basado en la matriz fundamental de dos cámaras y la resección del resto de cámaras. Para la estimación de la matriz fundamental se utiliza el algoritmo Gold Standard explicado en § 6.2.3.6 y para la resección también se emplea el respectivo Gold Standard § 5.4.2. Ambos algoritmos son rápidos. Además, se utiliza el ajuste de haces proyectivo (GS − Gold Standard) para optimizar dicha calibración inicial y se comparan los resultados de ambas reconstrucciones.

El criterio de comparación de los algoritmos es el error de reproyección, tanto residual (datos ruidosos) como de estimación (datos exactos), en función del número de puntos y de la varianza del ruido. Este criterio coincide con el coste propio que minimiza el ajuste de haces ($\boldsymbol{\epsilon}_{\text{res}}$) y es el perseguido desde la presentación de esta memoria (ver § 2.3).

Para las pruebas realizadas se han elegido valores de σ entre 0 y 5 en pasos de 1. El número de experimentos realizados es 100, suficientes para conocer la evolución del error. El número de puntos que se han elegido en las simulaciones es: $n = \{8, 10, 14, 30, 100\}$. El valor $n = 14$ es especialmente interesante, ya que en este punto son aproximadamente iguales el error residual y el error de estimación,

según las curvas teóricas (figura 6.31).

Comparación respecto del parámetro de calidad

Al comparar las gráficas de las pruebas experimentales (figuras 6.32 y 6.33) se aprecia el fenómeno de inversión: según crece el número de puntos n, disminuye la distancia a los datos exactos ($\boldsymbol{\epsilon}_{\text{est}}$) y aumenta la distancia a los datos ruidosos ($\boldsymbol{\epsilon}_{\text{res}}$). En las gráficas están señaladas con flechas las curvas con los valores medios. Para el algoritmo GS son prácticamente coincidentes las rectas con los valores medio (∘) y mediana (+), en cambio, las curvas con los valores medios del algoritmo INIC no coinciden con las curvas de las medianas, que son las curvas intermedias no apuntadas por ninguna flecha.

Vemos cómo, en la gráfica derecha de la figura 6.32, los errores residual y de estimación son aproximadamente iguales si se utilizan $n = 14$ puntos y $m = 10$ cámaras, según predice la teoría.

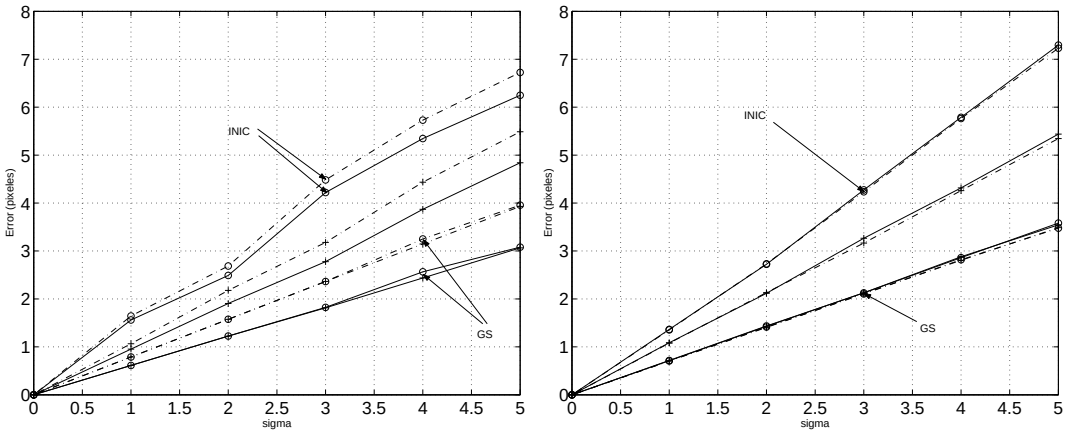


Figura 6.32: Geometría multicámara: valores RMS residual y de estimación del error de reproyección, para $n = 10$ puntos (izquierda) y $n = 14$ (derecha).

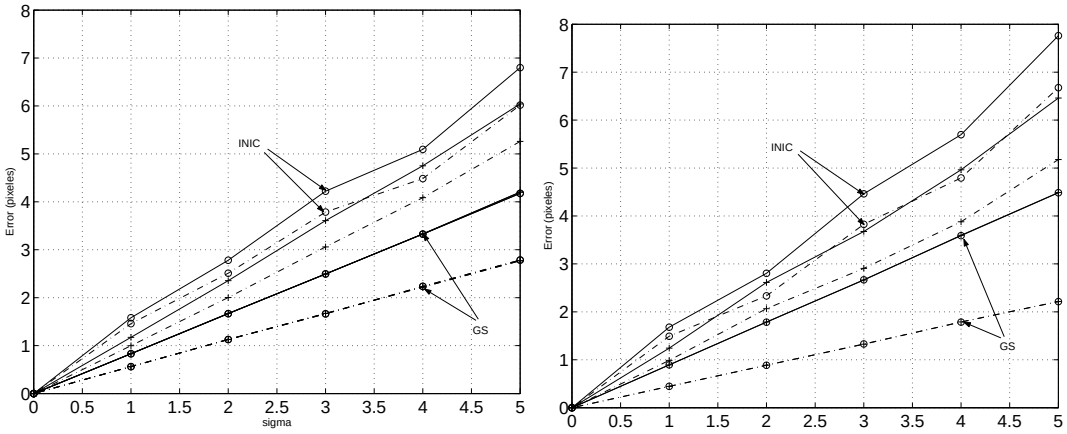


Figura 6.33: Geometría multicámara: valores RMS residual y de estimación del error de reproyección aproximado, para $n = 30$ puntos (izquierda) y $n = 100$ (derecha)

La conclusión que se debe sacar de estas gráficas es que el método de la calibración proyectiva inicial basada en la matriz fundamental seguida de la resección constituye una buena aproximación a la cal-

ibración proyectiva óptima. Esta aproximación facilita la optimización que realiza el ajuste de haces, que con pocas iteraciones del algoritmo de Levenberg-Marquardt converge a la solución que minimiza el error de reproyección.

Capítulo 7

Autocalibración

La autocalibración consiste en la obtención los parámetros de las cámaras dada una calibración proyectiva de las mismas, como ya se dijo en el capítulo de Introducción. Al contrario que la calibración, no se utiliza ningún patrón de calibración (ej. rejilla del espacio) ni objeto cuyas dimensiones sean conocidas, sólo la información contenida en las imágenes. Una vez que se conocen los parámetros intrínsecos de la cámara es posible obtener una reconstrucción métrica de la escena 3D.

El término "reconstrucción métrica" significa que difiere de la reconstrucción real en una transformación de semejanza del espacio, aunque a veces se confunde con el término "reconstrucción euclídea", que difiere de la real en una transformación euclídea.

La hipótesis de partida es que la cámara se mueve de forma rígida, (descrita mediante transformaciones del grupo euclídeo, contenido en el grupo de las transformaciones de semejanza) y esto implica que la cónica absoluta es fija ante transformaciones de semejanza, § 3.4.2. Lo mismo sucede con su dual, § 3.4.4. Si logramos identificar una cónica del espacio que es fija ante transformaciones de semejanza, ésta es Ω_∞, que define la estructura métrica del espacio.

En este capítulo primero se expone matemáticamente el objetivo del problema de autocalibración. A continuación se detallan las homografías entre las distintas reconstrucciones y se describen varios métodos de autocalibración, entre ellos: la hipótesis de cámaras ortogonales, las ecuaciones de Kruppa, las horópteras, etc.

7.1. Introducción

El dato principal de los algoritmos de autocalibración son las matrices de proyección de las cámaras $\mathtt{P}^i$ en una calibración proyectiva; no hacen falta los puntos 3D. El planteamiento general es el siguiente: existe una familia de homografías del espacio que hacen posible una descomposición de las matrices de proyección en la forma de cámaras proyectivas finitas: $\mathtt{P}^i\mathtt{H} = \mathtt{K}^i\mathtt{R}^i[\mathtt{I} \mid \mathbf{t}^i]$ que verifican restricciones adicionales. La matriz $\mathtt{H}$ se determina si hay suficientes datos y restricciones, ya que la mayoría de los algoritmos de autocalibración suponen algunas restricciones sobre los parámetros intrínsecos y/o extrínsecos para lograr su objetivo. Por ejemplo, es habitual la hipótesis de suponer que todas las cámaras son iguales, que comparten la misma matriz de parámetros intrínsecos $\mathtt{K}$, aunque sea desconocida, lo que hace que se necesiten menos vistas de las generales para estimar $\mathtt{K}$.

Siguiendo el esquema de procesamiento presentado en § 2.4 y dada la limitación de la calibración proyectiva, el objetivo es hallar una homografía $\mathtt{H}$ que convierta la reconstrucción proyectiva actual en una reconstrucción en un grupo de transformaciones más restringido: afín o conforme, conservando las proyecciones:

$$\{\mathtt{P}^i, \mathbf{X}_j\} \longrightarrow \{\mathtt{P}^i\mathtt{H}, \mathtt{H}^{-1}\mathbf{X}_j\}$$

$$\mathbf{x}_j^i \sim \mathtt{P}^i\mathbf{X}_j = (\mathtt{P}^i\mathtt{H})(\mathtt{H}^{-1}\mathbf{X}_j) = \hat{\mathtt{P}}^i\hat{\mathbf{X}}_j.$$

Existen dos formas de determinar $\mathtt{H}$: de una sola vez o en dos pasos (estratificadamente). En la calibración estratificada el paso costoso es el primero: determinar los elementos proyectivos de $\mathtt{H}$ (el plano del infinito π_∞); el resto se obtiene de forma lineal.

7.2. Conversiones entre reconstrucciones

Las dos siguientes secciones muestran cómo construir las dos homografías del espacio de la calibración estratificada: proyectivo $\longrightarrow$ afín $\longrightarrow$ métrico. En una sección posterior se hacen los dos pasos en uno solo.

7.2.1. Paso de una reconstrucción proyectiva a una afín

Para pasar de una calibración proyectiva a una calibración afín es necesario conocer la posición del plano de infinito π_∞.

Dadas las coordenadas duales de un plano, π, en $\mathbb{P}^3$, es inmediato obtener una homografía que lo transforma en el plano de coordenadas $\pi_\infty^e = (0:0:0:1)^\top$.

$$\mathtt{H}^{-\top}\pi = (0:0:0:1)^\top$$

Tal homografía es:

$$\mathtt{H} = \begin{bmatrix} \mathtt{I} \mid \mathbf{0} \\ \pi^\top \end{bmatrix}$$

Si $\pi = \pi_\infty$, después de aplicar esta homografía a las matrices de proyección de las cámaras y a los puntos 3D lo único que es seguro es que el plano del infinito está correctamente situado. La actual reconstrucción se diferencia de la real (euclídea) en una transformación proyectiva que deja fijo el plano del infinito, es decir, una transformación afín. Por eso se llama a la reconstrucción obtenida *reconstrucción afín*. Esta reconstrucción permite calcular cosas como: el punto medio entre dos puntos, el centroide de un conjunto de puntos, rectas o planos paralelos a uno dado, etc. Más referencias: capítulo 9, [1, pág. 250].

7.2.2. Paso de una reconstrucción afín a una euclídea o métrica

Una vez que el plano del infinito ha sido colocado en su posición canónica (euclídea π_∞^e) de tal forma que se dispone de una reconstrucción afín, entonces también se dispone de una aplicación entre imágenes, llamada la "homografía del infinito". Esta aplicación es una homografía de $\mathbb{P}^2$ que transfiere puntos de una imagen a otra a través del plano del infinito de la siguiente forma: se extiende el rayo reproyectado de un punto $\mathbf{x}$ en una imagen hasta cortar el plano del infinito en un punto $\mathbf{X}$; este punto se proyecta en el punto $\mathbf{x}'$ de la otra imagen. La homografía de $\mathbf{x}$ a $\mathbf{x}'$ se escribe: $\mathbf{x}' = \mathtt{H}_\infty\mathbf{x}$. Igual que en la figura 4.2, pero siendo $\pi = \pi_\infty$.

Tener una calibración afín es equivalente a conocer la homografía del infinito. Dadas dos cámaras $\mathtt{P} = [\mathtt{M} \mid \mathbf{m}]$ y $\mathtt{P}' = [\mathtt{M}' \mid \mathbf{m}']$ de una reconstrucción afín, la homografía del infinito está dada por la expresión $\mathtt{H}_\infty = \mathtt{M}'\mathtt{M}^{-1}$. Esto es así porque un punto $\mathbf{X} = (\tilde{\mathbf{X}}^\top, 0)^\top$ en el plano del infinito se transforma en el punto $\mathbf{x} = \mathtt{M}\tilde{\mathbf{X}}$ en una imagen y en $\mathbf{x}' = \mathtt{M}'\tilde{\mathbf{X}}$ en la otra, así que $\mathbf{x}' = \mathtt{M}'\mathtt{M}^{-1}\mathbf{x}$ para los puntos del plano del infinito. Además, se debe verificar que esto es invariante a una transformación afín del espacio aplicada a las cámaras. Por lo tanto, la homografía del infinito se puede calcular explícitamente dada una calibración afín y viceversa.

El paso que resta es transformar lo afín en lo métrico (determinar $\mathtt{K}$ dado π_∞). Este paso es mucho más fácil que el paso de lo proyectivo a lo afín. De hecho, existe una algoritmo lineal basado en la transformación de la IAC y su dual.

Para mejorar la calibración a una euclídea es necesario conocer la matriz de parámetros intrínsecos $\mathtt{K}$. Muchas veces se obtiene una reconstrucción euclídea identificando la DIAC $\omega^* = \mathtt{K}\mathtt{K}^\top$, la cónica de rectas dual de la proyección de la cónica absoluta. La cónica absoluta está en el plano del infinito, así que su proyección se mapea entre imágenes por medio de $\mathtt{H}_\infty$. Además, es invariante ante transformaciones de semejanzas (movimientos rígidos más escalados). Estas características producen las siguientes restricciones en la DIAC:

$$\omega^{*i} \sim \mathtt{H}^i_\infty \omega^* \mathtt{H}^{i\top}_\infty \tag{7.1}$$

siendo $\mathtt{H}^i_\infty$ la homografía de infinito entre la imagen de referencia y la i-ésima y ω^{*i} la DIAC de la i-ésima imagen.

$$\omega^i \sim (\mathtt{H}^i_\infty)^{-\top} \omega (\mathtt{H}^i_\infty)^{-1} \tag{7.2}$$

Las restricciones impuestas en una imagen se pueden transferir fácilmente a otra imagen y de esta forma juntarlas para estimar ω^* por métodos lineales. Una vez que se conoce ω^*, se obtiene $\mathtt{K}$ mediante la descomposición de Cholesky.

7.2.2.1. Solución para $\mathtt{K}$ constante

Si los parámetros intrínsecos son idénticos para todas las m imágenes, entonces $\mathtt{K}^i = \mathtt{K}$ y $\omega^{*i} = \omega^*$ para $i = 1, \ldots, m$, y la ec. (7.1) es

$$\omega^* \sim \mathtt{H}^i_\infty \omega^* \mathtt{H}^{i\top}_\infty \tag{7.3}$$

Es importante tener en cuenta los factores de escala en la anterior ecuación. Si normalizamos las homografías de tal forma que sus determinantes sean unitarios $\det \mathtt{H}^i_\infty = 1$, entonces se verifica la condición de igualdad.

$$\omega^* = \mathtt{H}^i_\infty \omega^* \mathtt{H}^{i\top}_\infty \tag{7.4}$$

De esta ecuación matricial se obtienen 6 ecuaciones independientes en los elementos de la matriz simétrica ω^*, para cada homografía $\mathtt{H}^i_\infty$. Entonces, podemos escribir (7.4) en forma de un sistema homogéneo de ecuaciones

$$\mathtt{A}_i \mathbf{c} = \mathbf{0}$$

siendo $\mathtt{A}_i$ una matriz de 6×6 construida a partir de $\mathtt{H}^i_\infty$ y $\mathbf{c} = [\omega^*_{11}, \omega^*_{12}, \omega^*_{13}, \omega^*_{22}, \omega^*_{23}, \omega^*_{33}]^\top$ es la DIAC, escrita en forma de vector 6×1. Si llamamos $B = \mathtt{H}^i_\infty$, entonces cada matriz $\mathtt{A}_i$ atiende a la siguiente expresión:

$$\mathtt{A}_i = \begin{bmatrix}
B_{11}^2 - 1 & 2B_{11}B_{12} & 2B_{11}B_{13} & B_{12}^2 & 2B_{12}B_{13} & B_{13}^2 \\
B_{21}B_{11} & B_{21}B_{12} + B_{22}B_{11} - 1 & B_{21}B_{13} + B_{23}B_{11} & B_{22}B_{12} & B_{22}B_{13} + B_{23}B_{12} & B_{23}B_{13} \\
B_{11}B_{31} & B_{11}B_{32} + B_{12}B_{31} & B_{11}B_{33} + B_{13}B_{31} - 1 & B_{12}B_{32} & B_{12}B_{33} + B_{13}B_{32} & B_{13}B_{33} \\
B_{21}^2 & 2B_{21}B_{22} & 2B_{21}B_{23} & B_{22}^2 - 1 & 2B_{22}B_{23} & B_{23}^2 \\
B_{21}B_{31} & B_{21}B_{32} + B_{22}B_{31} & B_{21}B_{33} + B_{23}B_{31} & B_{22}B_{32} & B_{22}B_{33} + B_{23}B_{32} - 1 & B_{23}B_{33} \\
B_{31}^2 & 2B_{31}B_{32} & 2B_{31}B_{33} & B_{32}^2 & 2B_{32}B_{33} & B_{33}^2 - 1
\end{bmatrix}$$

Con una sola ecuación matricial no es posible determinar $\mathbf{c}$ de forma única, pues $\mathtt{A}_i$ es a lo sumo de rango 4. Pero si se combinan ecuaciones de $m \geq 2$ imágenes, de tal forma que se construye una matriz $\mathtt{A}$ de $6m \times 6$ pegando las matrices $\mathtt{A}_i$ verticalmente y suponiendo que las rotaciones entre las imágenes son alrededor de distintos ejes, entonces en general $\mathbf{c}$ está determinada unívocamente.

La estabilidad numérica está relacionada con la unicidad. Si $\mathtt{H}_\infty$ no es muy precisa, no siempre es posible obtener una matriz ω (o ω^*) definida positiva, por lo que después no podemos aplicar la descomposición de Cholesky para obtener $\mathtt{K}$. Esta sensibilidad se reduce si se proporcionan múltiples movimientos y se estima ω combinando restricciones de múltiples homografías del infinito.

Las referencias de este apartado son [6], [27] y § 18.5.2 de [1].

7.2.3. Paso de una reconstrucción proyectiva a una euclídea o métrica

Para pasar de una calibración proyectiva a una calibración euclídea es necesario conocer la posición del plano de infinito π_∞ y la cónica absoluta Ω_∞.

RESULTADO

Una calibración proyectiva $\{\mathtt{P}^i, \mathbf{X}_j\}$ en la que $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$ se puede convertir en una calibración euclídea $\{\mathtt{P}^i\mathtt{H}, \mathtt{H}^{-1}\mathbf{X}_j\}$ mediante una matriz $\mathtt{H}$ de la forma

$$\mathtt{H} = \begin{bmatrix} \mathtt{K} & \mathbf{0} \\ -\mathbf{p}^\top \mathtt{K} & 1 \end{bmatrix} \tag{7.5}$$

donde $\mathtt{K}$ es una matriz triangular superior. Más aún:

1. $\mathtt{K} = \mathtt{K}^1$ es la matriz de parámetros intrínsecos de la primera cámara.
2. Las coordenadas del plano del infinito en la calibración proyectiva son $\pi_\infty = (\mathbf{p}^\top, 1)^\top$.

Por tanto, si conocemos el plano del infinito en la referencia proyectiva y la matriz de parámetros intrínsecos de la primera cámara, entonces la transformación $\mathtt{H}$ que convierte la reconstrucción proyectiva en una euclídea viene dada por (7.5).

DEMO

Supongamos que tenemos una calibración proyectiva $\{\mathtt{P}^i, \mathbf{X}_j\}$; entonces, basándonos en las restricciones de los parámetros intrínsecos de las cámaras o restricciones en el movimiento (parámetros extrínsecos) queremos determinar una homografía $\mathtt{H}$ que rectifique la calibración, tal que $\{\mathtt{P}^i\mathtt{H}, \mathtt{H}^{-1}\mathbf{X}_j\}$ sea una calibración euclídea.

Partamos de una verdadera situación métrica con cámaras calibradas, y la escena representada en una referencia euclídea. Tenemos m cámaras $\mathtt{P}^i_{\mathrm{M}}$ que proyectan un punto 3D $\mathbf{X}_{\mathrm{M}}$ en un punto de la imagen $\mathbf{x}^i = \mathtt{P}^i_{\mathrm{M}}\mathbf{X}_{\mathrm{M}}$; uno por cada imagen. El subíndice M indica que las cámaras están calibradas y que la referencia espacial es la euclídea. Podemos escribir las cámaras como: $\mathtt{P}^i_{\mathrm{M}} = \mathtt{K}^i[\mathtt{R}^i \mid \mathbf{t}^i]$, para $i = 1, \ldots, m$.

En una calibración proyectiva obtenemos las cámaras $\mathtt{P}^i$ que están relacionadas con $\mathtt{P}^i_{\mathrm{M}}$ según:

$$\mathtt{P}^i_{\mathrm{M}} = \mathtt{P}^i\mathtt{H} \qquad i = 1, \ldots, m \tag{7.6}$$

donde $\mathtt{H}$ es una homografía del espacio, 4×4, desconocida. El objetivo es determinar $\mathtt{H}$. Para ser precisos, podremos obtener una $\mathtt{H}$ que reconstruya salvo una semejanza: no podemos recuperar los valores absolutos de la rotación, traslación y escalado. Elegimos que el sistema de referencia coincida con el de la primera cámara, tal que $\mathtt{R}^1 = \mathtt{I}$ y $\mathbf{t}^1 = \mathbf{0}$. Entonces $\mathtt{R}^i$,$\mathbf{t}^i$ especifican las transformaciones euclídeas entre la i-ésima cámara y la primera, y $\mathtt{P}^1_{\mathrm{M}} = \mathtt{K}^1[\mathtt{I} \mid \mathbf{0}]$. Del mismo modo, en la calibración proyectiva elegimos la matriz de proyección canónica para la primera cámara, es decir, $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$. Escribiendo $\mathtt{H}$ de la forma

$$\mathtt{H} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{v}^\top & k \end{bmatrix}$$

la condición $\mathtt{P}^1_{\mathrm{M}} = \mathtt{P}^1\mathtt{H}$ de (7.6) viene a ser $[\mathtt{K} \mid \mathbf{0}] = [\mathtt{I} \mid \mathbf{0}]\mathtt{H}$, lo que implica que $\mathtt{A} = \mathtt{K}^1$ y $\mathbf{t} = \mathbf{0}$. Además, como $\mathtt{H}$ es no singular, k debe ser no nulo, así que podemos suponer $k = 1$ (esto fija la escala de la reconstrucción). Esto muestra que $\mathtt{H}$ es de la forma

$$\mathtt{H} = \begin{bmatrix} \mathtt{K}^1 & \mathbf{0} \\ \mathbf{v}^\top & 1 \end{bmatrix}$$

El vector $\mathbf{v}$, junto con $\mathtt{K}^1$, especifica el plano del infinito en la reconstrucción proyectiva, ya que las coordenadas de π_∞ son

$$\pi_\infty = \mathtt{H}^{-\top} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{bmatrix} (\mathtt{K}^1)^{-\top} & -(\mathtt{K}^1)^{-\top}\mathbf{v} \\ \mathbf{0} & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -(\mathtt{K}^1)^{-\top}\mathbf{v} \\ 1 \end{pmatrix}$$

Escribiremos $\pi_\infty = (\mathbf{p}^\top, 1)^\top$ para que $\mathbf{p} = -(\mathtt{K}^1)^{-\top}\mathbf{v}$.

Del resultado anterior se deduce que para transformar una reconstrucción proyectiva en una euclídea basta con especificar 8 parámetros - los tres elementos de $\mathbf{p}$ y los cinco elementos de $\mathtt{K}^1$. Esto concuerda con la cuenta geométrica. Encontrar la estructura métrica es equivalente a especificar el plano del infinito y la cónica absoluta (3 y 5 grados de libertad respectivamente). En una reconstrucción métrica, la calibración $\mathtt{K}^i$ de cada cámara, su rotación $\mathtt{R}^i$ relativa a la primera cámara, y su traslación $\mathbf{t}^i$, relativa a la primera cámara salvo un factor de proporcionalidad común, $\mathbf{t}^i \mapsto s\mathbf{t}^i$, están todos determinados.

7.3. Cámaras ortogonales

Como se comentó en § 2.5, la hipótesis de cámaras ortogonales consiste en suponer que tienen píxeles cuadrados y punto principal en el centro en el origen de coordenadas. Sólo queda determinar la distancia focal, $\alpha = \alpha_u = \alpha_v$:

$$\mathtt{K} = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

El algoritmo de estimación de la distancia focal y el plano del infinito consiste en utilizar las propiedades de la cuádrica absoluta dual $\mathtt{Q}^*_\infty$ respecto del ángulo formado por dos planos, § 3.4.4. En concreto, recordemos que dos planos son ortogonales sii son conjugados respecto de $\mathtt{Q}^*_\infty$, ecuación (3.14): $\pi_1^\top \mathtt{Q}^*_\infty \pi_2 = 0$.

Hay tres planos que son mutuamente ortogonales en una cámara como la considerada: el plano principal y los planos axiales, § 5.3. Por eso se llama a estas cámaras "ortogonales". Los distintos planos son las filas de la matriz de proyección:

$$\mathtt{P} = \begin{bmatrix} \mathbf{P}^{1\top} \\ \mathbf{P}^{2\top} \\ \mathbf{P}^{3\top} \end{bmatrix} = \begin{bmatrix} \pi_1^\top \\ \pi_2^\top \\ \pi_3^\top \end{bmatrix}$$

Y las ecuaciones de autocalibración consisten en estimar la cuádrica dual que mejor se ajusta a las condiciones $\pi_i^\top \mathtt{Q}^*_\infty \pi_j = 0, \forall i \neq j$. El problema de optimización que se plantea para m cámaras ortogonales es:

$$\min_{\mathtt{Q}^*} \sum_{k=1}^{m} \sum_{\substack{i,j=1 \\ i \neq j}}^{3} (\pi_i^{k\top} \mathtt{Q}^* \pi_j^k)^2$$

Las condiciones $\pi_i^{k\top} \mathtt{Q}^* \pi_j^k$ son lineales en los elementos de $\mathtt{Q}^*$, y se pueden expresar de la siguiente forma:

$$\pi_i^\top \mathtt{Q}^* \pi_j = \mu_{\pi_i \pi_j}^\top \mathbf{q}^* = 0 \quad \forall (i,j)$$

como se explica en la ec. (7.36) de § 7.8.6. Además, imponemos la restricción $\|\mathtt{Q}^*\|_F = 1 \Leftrightarrow \|\mathbf{q}^*\| = 1$. Entonces, podemos expresar el problema de minimización:

$$\min_{\mathtt{Q}^*} \sum_{k=1}^{m} \sum_{\substack{i,j=1 \\ i \neq j}}^{3} (\pi_i^{k\top} \mathtt{Q}^* \pi_j^k)^2 \quad \text{sujeto a} \quad \|\mathtt{Q}^*\|_F = 1 \Leftrightarrow \min_{\mathbf{q}^*} \|\mathtt{A}\mathbf{q}^*\|^2 \quad \text{sujeto a} \quad \|\mathbf{q}^*\| = 1$$

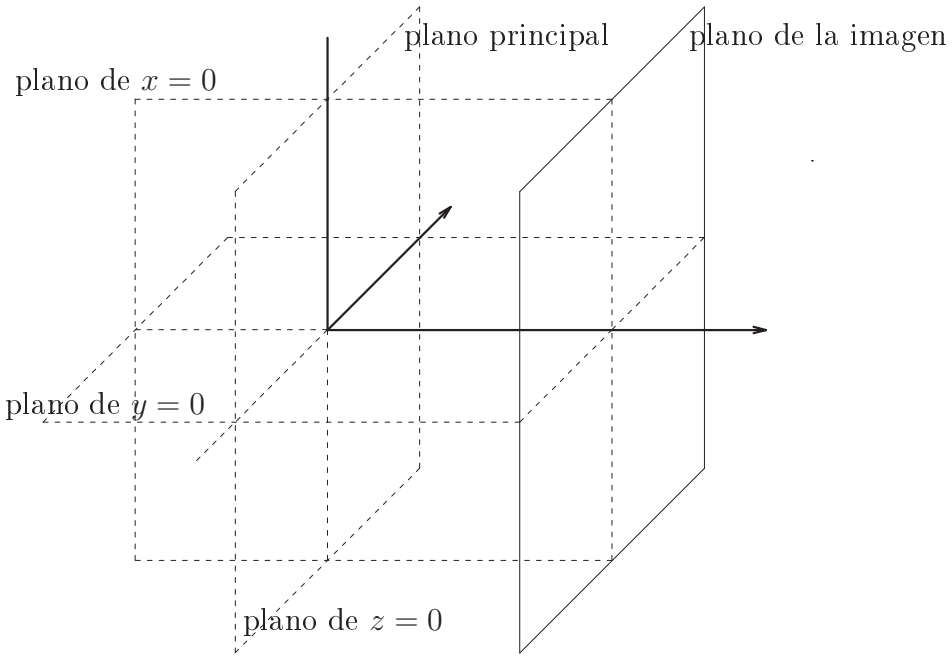


Figura 7.1: Cámaras ortogonales

donde

$$\mathtt{A} = \begin{bmatrix} \mathtt{A}_1 \\ \vdots \\ \mathtt{A}_m \end{bmatrix} \qquad \mathtt{A}_i = \begin{bmatrix} \mu_{\pi_1}^\top \pi_2 \\ \mu_{\pi_1}^\top \pi_3 \\ \mu_{\pi_2}^\top \pi_3 \end{bmatrix}$$

es la matriz de diseño. Esto no es otra cosa que minimizar la distancia algebraica $\|\epsilon\| = \|\mathtt{A}\mathbf{q}^*\|$, habitual de los algoritmos lineales. La solución es archiconocida: la mínima distancia algebraica es el menor valor singular de $\mathtt{A}$ (o la raíz cuadrada del menor autovalor de $\mathtt{A}^\top\mathtt{A}$) y la cuádrica $\mathbf{q}^*$ en forma de vector es el vector singular (derecho) asociado al menor valor singular (el autovector asociado al menor autovalor).

Este algoritmo proporciona una cuádrica que no tiene porqué ser semejante a la cuádrica absoluta dual, ya que no han sido impuestas las restricciones necesarias. Sin embargo, se puede obtener una estimación del plano del infinito como el autovector asociado al menor autovalor. En este caso, $\mathtt{Q}^*\boldsymbol{\pi} = \lambda_4\boldsymbol{\pi}$. El coste asociado a esta aproximación es $\|\mathtt{Q}^*\boldsymbol{\pi}\| = |\lambda_4|$, que en el caso ideal (si $\mathtt{Q}^* = \mathtt{Q}^*_\infty$) vale $|\lambda_4| = 0$, ya que la cuádrica absoluta dual es degenerada.

Para cada cámara, podemos hallar la imagen de la cuádrica estimada, las que consideraríamos que son las DIACs, mediante la ecuación $\omega^{*i} = \mathtt{P}^i\mathtt{Q}^*_\infty\mathtt{P}^{i\top}$. Y una vez conocida la DIAC, utilizar la descomposición de Cholesky para hallar las matrices de parámetros intrínsecos: $\omega^{*i} = \mathtt{K}^i\mathtt{K}^{i\top}$.

El algoritmo queda resumido en los siguientes pasos: dadas m matrices de proyección $\mathtt{P}^i, i = 1..m$:

1. Estimar la cuádrica absoluta dual:
   - *a*) Calcular la matriz de diseño $\mathtt{A}$ concatenando las matrices de diseño de cada cámara $\mathtt{A}_i$
   - *b*) Hallar la SVD de $\mathtt{A} = \mathtt{UDV}^\top$
   - *c*) La mínima distancia algebraica es: $\mathtt{coste} = \sigma_n = \text{mín}\{\text{diag}(\mathtt{D})\}$
   - *d*) La cuádrica estimada es $\mathbf{q}^* = \mathtt{V}(:, n)$. Pasar de la notación de vectorial a la matricial $\mathbf{q}^* \mapsto \mathtt{Q}^*$.

2. Hallar el plano del infinito como el autovector asociado al menor autovalor de $\mathtt{Q}^*$. Si se desea se puede proyectar la cuádrica $\mathtt{Q}^*$ sobre el espacio de las cuádricas de rango 3, anulando el menor autovalor.

3. Para cada cámara hallar la DIAC: $\omega^{*i} = \mathtt{P}^i \mathtt{Q}^*_\infty \mathtt{P}^{i\top}$

4. Hallar las matrices de parámetros intrínsecos mediante la descomposición de Cholesky: $\omega^{*i} = \mathtt{K}^i \mathtt{K}^{i\top}$

La matriz $\mathtt{A}$ tiene 10 columnas y el sistema homogéneo $\mathtt{A}\mathbf{q}^* = \mathbf{0}$ tiene solución exacta sii $\mathtt{A}$ es de rango 9, que es la situación si hay tres cámaras, entonces $n = 9$. Si hay $m \geq 4$ cámaras, puede no existir solución exacta y por eso se busca la solución mínimo-cuadrática, $n = 10$.

7.4. Ecuaciones Kruppa

Las ecuaciones de Kruppa fueron el primer método de autocalibración conocido. Son restricciones en dos imágenes para las que sólo hace falta conocer la matriz fundamental $\mathtt{F}$ y consisten en dos ecuaciones cuadráticas independientes para los elementos de la DIAC, ω^*.

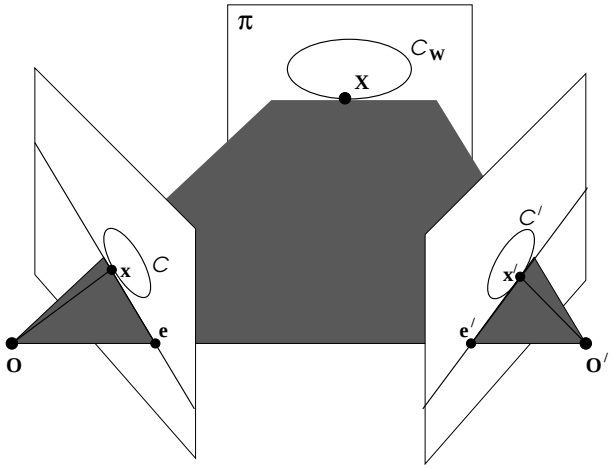


Figura 7.2: Ecuaciones de Kruppa: tangencia epipolar a una cónica

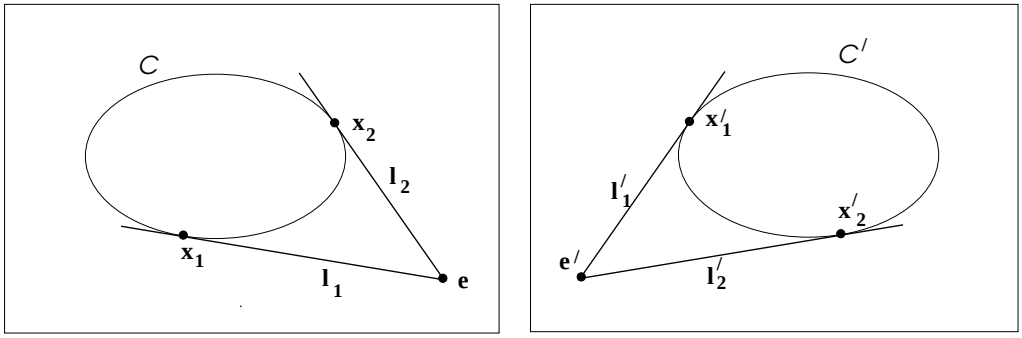


Figura 7.3: Ecuaciones de Kruppa: proyecciones de una cónica y sus correspondencias de rectas epipolares tangentes

Las ecuaciones son la representación algebraica de las rectas epipolares correspondientes que son tangentes a una cónica. La figuras 7.2 y 7.3 ilustran la geometría de esta correspondencia. Una cónica

$\mathtt{C}_W$ en un plano del espacio π se proyecta en un par de cónicas correspondientes $\mathtt{C} \leftrightarrow \mathtt{C}'$, una en cada imagen. Los dos planos que pasan por los centros ópticos y son tangentes a la cónica $\mathtt{C}_W$ cortan a los planos de las imágenes en sendas correspondencias de rectas epipolares tangentes a las cónicas $\mathtt{C}$ y $\mathtt{C}'$. Las rectas $\mathbf{l}_1 = \mathbf{e} \times \mathbf{x}_1 = [\mathbf{e}]_\times \mathbf{x}_1$ y $\mathbf{l}_2 = \mathbf{e} \times \mathbf{x}_2 = [\mathbf{e}]_\times \mathbf{x}_2$ se pueden combinar en una sola cónica degenerada de puntos, $\mathtt{C}_t = [\mathbf{e}]_\times \mathtt{C}^* [\mathbf{e}]_\times$, siendo $\mathtt{C}^*$ la dual de $\mathtt{C}$. Lo mismo se puede hacer en la segunda imagen y la cónica degenerada es $\mathtt{C}'_t = [\mathbf{e}']_\times \mathtt{C}^{*'} [\mathbf{e}']_\times$. Las rectas epipolares son correspondientes según marca la homografía inducida por el plano $\boldsymbol{\pi}$ y según dicha homografía, la cónica $\mathtt{C}_t$ se transforma en $\mathtt{C}'_t = \mathtt{H}^{-\top} \mathtt{C}_t \mathtt{H}^{-1}$. Por último, la matriz fundamental es $\mathtt{F} = \mathtt{H}^{-\top} [\mathbf{e}]_\times$.

El caso interesante sucede cuando la cónica $\mathtt{C}_W$ es la cónica absoluta Ω_∞. Entonces $\mathtt{C}^* = \omega^*$, $\mathtt{C}'^* = \omega^{*'}$, $\mathtt{H} = \mathtt{H}_\infty$ e igualando las expresiones para $\mathtt{C}'_t$, se llega a la ecuación

$$[\mathbf{e}']_\times \omega^{*'} [\mathbf{e}']_\times \sim \mathtt{F} \omega^* \mathtt{F}^\top \tag{7.7}$$

Si las dos cámaras consideradas tienen los mismos parámetros intrínsecos, entonces $\omega^{*'} = \omega^*$ y las ecuaciones de Kruppa son $[\mathbf{e}']_\times \omega^* [\mathbf{e}']_\times = \mathtt{F} \omega^* \mathtt{F}^\top$. Al ser una igualdad salvo proporcionalidad, se obtienen ecuaciones cuadráticas en los elementos de ω^*.

Las ecuaciones de Kruppa se pueden expresar de una forma más cómoda como un producto vectorial.

$$\begin{pmatrix} \mathbf{u}_2^\top \omega^{*'} \mathbf{u}_2 \\ -\mathbf{u}_1^\top \omega^{*'} \mathbf{u}_2 \\ \mathbf{u}_1^\top \omega^{*'} \mathbf{u}_1 \end{pmatrix} \times \begin{pmatrix} \sigma_1^2 \mathbf{v}_1^\top \omega^* \mathbf{v}_1 \\ \sigma_1 \sigma_2 \mathbf{v}_1^\top \omega^* \mathbf{v}_2 \\ \sigma_2^2 \mathbf{v}_2^\top \omega^* \mathbf{v}_2 \end{pmatrix} = \mathbf{0} \tag{7.8}$$

donde $\mathbf{u}_i$, $\mathbf{v}_i$ y σ_i son las columnas y los valores singulares de la SVD de $\mathtt{F}$. Esto da lugar a tres ecuaciones cuadráticas en los elementos ω^*_{ij} de ω^*, dos de las cuales son independientes.

DEMO: La matriz fundamental tiene rango 2 y su SVD es $\mathtt{F} = \mathtt{U}\,\mathrm{diag}(\sigma_1, \sigma_2, 0)\mathtt{V}^\top$. Los epipolos son $\mathbf{e} = \mathbf{v}_3$ y $\mathbf{e}' = \mathbf{u}_3$. Sustituyamos en (7.7).

$$\begin{aligned}
[\mathbf{u}_3]_\times \omega^{*'} [\mathbf{u}_3]_\times &\sim \mathtt{U}\mathtt{D}\mathtt{V}^\top \omega^* \mathtt{V}\mathtt{D}\mathtt{U}^\top \\
\mathtt{U}^\top [\mathbf{u}_3]_\times \omega^{*'} [\mathbf{u}_3]_\times \mathtt{U} &\sim \mathtt{D}\mathtt{V}^\top \omega^* \mathtt{V}\mathtt{D} \\
[\mathbf{u}_2 \ -\mathbf{u}_1 \ 0] \omega^{*'} [\mathbf{u}_2 \ -\mathbf{u}_1 \ 0] &\sim \begin{bmatrix} \sigma_1\mathbf{v}_1 & \sigma_2\mathbf{v}_2 & \mathbf{e} \end{bmatrix}^\top \omega^* \begin{bmatrix} \sigma_1\mathbf{v}_1 & \sigma_2\mathbf{v}_2 & \mathbf{e} \end{bmatrix} \\
\begin{bmatrix} \mathbf{u}_2^\top \omega^{*'} \mathbf{u}_2 & -\mathbf{u}_2^\top \omega^{*'} \mathbf{u}_1 & 0 \\ -\mathbf{u}_1^\top \omega^{*'} \mathbf{u}_2 & \mathbf{u}_1^\top \omega^{*'} \mathbf{u}_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} &\sim \begin{bmatrix} \sigma_1^2 \mathbf{v}_1^\top \omega^* \mathbf{v}_1 & \sigma_1\sigma_2 \mathbf{v}_1^\top \omega^* \mathbf{v}_2 & 0 \\ \sigma_1\sigma_2 \mathbf{v}_1^\top \omega^* \mathbf{v}_2 & \sigma_2^2 \mathbf{v}_2^\top \omega^* \mathbf{v}_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}
\end{aligned}$$

Esta igualdad salvo proporcionalidad se puede expresar según (7.8), ya que sólo 3 de los 4 elementos no nulos en cada matriz son distintos entre sí.

Un algoritmo de autocalibración inspirado en las ecuaciones de Kruppa para varias cámaras puede suponer que todas tienen los mismos parámetros intrínsecos y tratar de optimizar la distancia algebraica de este problema: cómo de bien se verifican las ecuaciones de Kruppa. En presencia de ruido se puede optimizar la suma de módulos al cuadrado del producto vectorial (7.8), esto es:

$$\min_{\omega^*} \sum_k \|\epsilon_k\|^2 = \min_{\omega^*} \sum_k \left\| \begin{pmatrix} \mathbf{u}_{2k}^\top \omega^* \mathbf{u}_{2k} \\ -\mathbf{u}_{1k}^\top \omega^* \mathbf{u}_{2k} \\ \mathbf{u}_{1k}^\top \omega^* \mathbf{u}_{1k} \end{pmatrix} \times \begin{pmatrix} \sigma_{1k}^2 \mathbf{v}_{1k}^\top \omega^* \mathbf{v}_{1k} \\ \sigma_{1k}\sigma_{2k} \mathbf{v}_{1k}^\top \omega^* \mathbf{v}_{2k} \\ \sigma_{2k}^2 \mathbf{v}_{2k}^\top \omega^* \mathbf{v}_{2k} \end{pmatrix} \right\|^2$$

donde k recorre el número de parejas de cámaras para las que se conoce la matriz fundamental $\mathtt{F}$.

Este problema admite una formulación en términos de una función modelo, así que es aplicable es esquema general del algoritmo LM, que funciona muy bien. El vector de parámetros de la función modelo es la DIAC, de 6 elementos (en forma de vector). Además se deben proporcionar las matrices fundamentales de las parejas de cámaras. El vector de medidas es tal que su módulo al cuadrado es la función de coste, ya que el vector de medidas objetivo es el nulo.

$$f : \omega^* \longrightarrow \epsilon = \begin{bmatrix} \epsilon_1^\top & \cdots & \epsilon_k^\top & \cdots \end{bmatrix}^\top$$

Para que la función de coste trate por igual a todos los pares de cámaras con independencia del factor de proporcionalidad de la matriz fundamental es necesario que todas las matrices fundamentales estén normalizadas a la misma norma Frobenius, por ejemplo la unidad, así los valores singulares están todos balanceados. Lo mismo es aplicable a la DIAC, antes de aplicar el producto vectorial (7.8) se debe normalizar la DIAC, por ejemplo a norma unidad, también. Con estas dos precauciones la función de coste es invariante al escalado.

Las ecuaciones de Kruppa exhiben algunos problemas: movimientos degenerados (si no hay rotación entre las cámaras, entonces las ecuaciones de Kruppa no aportan ninguna restricción) y estimaciones menos fiables que otros métodos basados en otros conceptos que impongan restricciones. Por ejemplo el método de las ecuaciones de Kruppa con varias imágenes no impone que el plano soporte para la cónica del espacio de la cual se estiman sus imágenes sea único. Por eso se estudian otros métodos de estimación.

7.5. La restricción unimodular

El plano del infinito para cámaras con mismos parámetros intrínsecos y movimientos no degenerados se puede hallar a través de la restricción unimodular, ecuación polinómica en las coordenadas de π_∞.

La restricción unimodular afirma que la homografía de infinito $\mathtt{H}_\infty^i$ de cámaras con idénticos parámetros intrínsecos es conjugada de una rotación, salvo proporcionalidad, por lo tanto todos sus autovalores son del mismo módulo. De hecho se cumple que $\mathtt{H}_\infty^i = \mu\mathtt{K}\mathtt{R}\mathtt{K}^{-1}$, así que los autovalores son $\{\mu, \mu e^{i\theta}, \mu e^{-i\theta}\}$, como corresponde a una rotación y escalado, § 5.2.1.

Para hallar los autovalores hay que calcular el polinomio característico de la matriz $\mathtt{H}_\infty^i$, que contiene una parametrización del plano del infinito, π_∞.

$$\det(\mathtt{H}_\infty^i - \lambda\mathtt{I}) = a\lambda^3 + b\lambda^2 + c\lambda + d$$

La restricción unimodular impone que las raíces de esta ecuación sean del mismo módulo: $|\lambda_1| = |\lambda_2| = |\lambda_3| (= \mu)$. Desarrollando la expresión anterior, se llega a la ecuación:

$$ac^3 = b^3 d$$

La restricción unimodular se puede escribir como una ecuación de grado 4 en los elementos del plano del infinito. Cada par de cámaras genera una ecuación de este tipo. Buscando las soluciones de estas ecuaciones se obtiene π_∞. Para más detalles, se remite al lector a las referencias [27] y [1, pág. 458].

La función de coste del algoritmo que se ha desarrollado se resume en los siguientes pasos: dadas m matrices de proyección $\mathtt{P}^i$ y un plano candidato a ser plano del infinito,

1. Calcular la homografía de infinito de cada cámara, $\mathtt{H}_\infty^i$.
2. Hallar los autovalores de $\mathtt{H}_\infty^i$.
3. Calcular un coste sobre la relación de módulos de los autovalores. Sumar todos los costes parciales.

La función de coste, inspirada en [9] al igual que se explica en § 7.22, asigna a un **plano** candidato el valor:

$$c_1(\textbf{plano}) = \sum_{k=1}^{m} \sum_{\substack{p,q=1 \\ p\neq q}}^{3} \left(\left| \frac{\lambda_p^k}{\lambda_q^k} \right| - 1 \right)^2 \tag{7.9}$$

Si se expande la expresión anterior para mayor claridad, se obtienen 6 términos para cada k, como en (7.23). Este coste es invariante a escalado y admite una función modelo, similar a la expresada en (7.8.4.1).

$$f : \textbf{plano} \equiv \mathbf{P} \rightarrow \widehat{\mathbf{X}} \equiv \left[\widehat{\mathbf{X}}_1^\top \ \cdots \ \widehat{\mathbf{X}}_k^\top \ \cdots \ \widehat{\mathbf{X}}_m^\top \right]^\top$$

con

$$\widehat{\mathbf{X}}_k = \left[\left| \frac{\lambda_1^k}{\lambda_2^k} \right| - 1, \left| \frac{\lambda_1^k}{\lambda_3^k} \right| - 1, \ \left| \frac{\lambda_2^k}{\lambda_1^k} \right| - 1, \ \left| \frac{\lambda_3^k}{\lambda_1^k} \right| - 1, \ \left| \frac{\lambda_2^k}{\lambda_3^k} \right| - 1, \ \left| \frac{\lambda_3^k}{\lambda_2^k} \right| - 1 \right]^\top \tag{7.10}$$

El vector de medidas objetivo es el nulo, ya que la solución se obtiene cuando el coste vale cero.

Este algoritmo de la restricción unimodular es, como muchos algoritmos, sesgado en la primera cámara porque se suele elegir que la primera matriz de proyección sea $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$, como en § 7.2.3. Así se simplifica la homografía que actualización la reconstrucción.

7.6. Cuádrica Absoluta Dual

7.6.1. Algoritmo 1

El algoritmo de Triggs [15] es un clásico para estimar la cuádrica absoluta dual $\mathtt{Q}_\infty^*$ y la DIAC. Se basa en la propiedad de proyección de la cuádrica absoluta dual en el plano de la imagen:

$$\omega^{*i} \sim \mathtt{P}^i \mathtt{Q}_\infty^* \mathtt{P}^{i\top} \tag{7.11}$$

En la citada referencia se proponen dos algoritmos para cámaras con los mismos parámetros intrínsecos: uno cuasi-lineal que requiere $m = 4$ cámaras y otro no lineal, que funciona con $m = 3$ o más cámaras; pero ambos se basan en el mismo principio.

Si se desea explotar la ecuación (7.6) hay que eliminar el factor de proporcionalidad. En los algoritmos de cálculo de homografías y de la matriz de proyección lo conseguíamos por medio del producto vectorial $\mathbf{x} \sim \mathtt{P}\mathbf{X} \Rightarrow \mathbf{x} \times \mathtt{P}\mathbf{X} = \mathbf{0}$. En este caso aplicamos el mismo concepto:

$$\omega^* \times (\mathtt{P}^i \mathtt{Q}_\infty^* \mathtt{P}^{i\top}) = \mathbf{0} \tag{7.12}$$

sin embargo, no tiene significado de producto vectorial ya que el producto vectorial es una operación que sólo existe para vectores en $\mathbb{R}^3$, aunque la llamaremos así. Desarrollando la anterior ecuación en función de los elementos de las matrices simétricas 3×3 que intervienen, se llega a:

$$\omega^{*AB}(\mathtt{P}^i \mathtt{Q}_\infty^* \mathtt{P}^{i\top})^{CD} - \omega^{*CD}(\mathtt{P}\mathtt{Q}_\infty^* \mathtt{P}^\top)^{AB} = 0 \tag{7.13}$$

Cada cámara proporciona $\binom{6}{2} = 15$ ecuaciones bilineales (5 linealmente independientes) en los $10+6 = 16$ elementos de ω^* y $\mathtt{Q}_\infty^*$, cuyos coeficientes son cuadráticos en los elementos de las matrices de proyección (datos).

Dadas m cámaras, ambos algoritmos utilizan las $15m$ ecuaciones, para evitar elegir 5 por cámara y que se produzcan casos degenerados. La solución es minimizar el coste algebraico que indica cómo se verifican las ecuaciones. El método no lineal utiliza algoritmos de optimización numérica con restricciones, mientras que el algoritmo lineal utiliza una factorización basada en la SVD. Sólo el método no lineal impone la condición de rango 3 de la cuádrica absoluta dual ($\det \mathtt{Q}_\infty^* = 0$) mientras se realiza la optimización, lo que le confiere mayor estabilidad que el método lineal. En el algoritmo lineal, es

posible imponer esta condición a posteriori, según se ha explicado en § 7.3.

Una vez que se conoce la DIAC, es inmediato calcular la matriz de parámetros intrínsecos $\mathtt{K}$ mediante la descomposición de Cholesky. También es posible construir la homografía que actualiza la reconstrucción en una de semejanza a partir de la descomposición en autovalores y autovectores de la matriz de la cuádrica absoluta dual.

$$\mathtt{Q}_\infty^* \sim \mathtt{V}\mathbf{\Lambda}\mathtt{V}^\top \sim \mathtt{H}\,\mathrm{diag}(1,1,1,0)\mathtt{H}^\top \tag{7.14}$$

donde $\mathtt{H} = \mathtt{V}\mathbf{\Lambda}^{1/2}$ es el cambio que hay que aplicar: $\mathtt{P}_\mathrm{M} = \mathtt{P}\mathtt{H}$ y $\mathbf{X}_\mathrm{M} = \mathtt{H}^{-1}\mathbf{X}$. Las columnas de $\mathtt{H}$ constituyen una base euclídea en coordenadas homogéneas (3 direcciones ortogonales y un origen de coordenadas).

Algoritmo lineal

En presencia de ruido la ecuación (7.12) no se verificará de forma exacta, sino con cierto error, $\epsilon_i = \omega^* \times (\mathtt{P}^i\mathtt{Q}_\infty^*\mathtt{P}^{i\top}) \in \mathbb{R}^{15}$. Parece lógico plantear el problema de minimizar la norma del vector de error (para todas las cámaras) por mínimos cuadrados.

$$\min_{\omega^*,\mathtt{Q}_\infty^*} \sum_{i=1}^{m} \|\epsilon_i\|^2 = \sum_{i=1}^{m} \|\omega^* \times (\mathtt{P}^i\mathtt{Q}_\infty^*\mathtt{P}^{i\top})\|^2$$

Además, para que la anterior función de coste sea invariante al escalado, se deben normalizar (por ejemplo a norma Frobenius unidad) las matrices ω^*, $\mathtt{Q}_\infty^*$ y $\mathtt{P}^i$.

Como ω^* y $\mathtt{Q}_\infty^*$ son matrices simétricas, podemos quedarnos con las partes triangulares superiores sin perder información y expresarlas en forma de vectores $\mathbf{q}^* \in \mathbb{R}^{10}$ y $\boldsymbol{\omega} \in \mathbb{R}^6$ (de ahí los números 6 y 10). Como bien se ha indicado, las ecuaciones (7.13) son bilineales en esos 16 elementos que forman la ω^* y $\mathtt{Q}_\infty^*$. Esto significa que se pueden expresar de forma lineal en los elementos del producto tensorial de las dos matrices: $\mathtt{Q}^* \otimes \omega^*$, que lo expresamos mediante el producto matricial de un vector columna por un vector fila: $\mathtt{G} = \mathbf{q}^*\boldsymbol{\omega}^\top$ de dimensiones 10×6. Esta matriz se puede reordenar en forma de vector $\mathbf{g} \in \mathbb{R}^{60}$, de tal forma que las ecuaciones (7.13) se expresan en forma estándar $\mathtt{A}\mathbf{g} = 0$, donde $\mathtt{A}$ es la matriz de diseño. Como bien se ha dicho y es habitual, si hay ruido el problema se transforma en:

$$\min_{\mathbf{g}} \|\mathtt{A}\mathbf{g}\|^2 \quad \text{sujeto a} \quad \|\mathbf{g}\| = 1$$

Este es el problema clásico, de los pocos que sabemos resolver, cuya solución se obtiene mediante la SVD de $\mathtt{A} = \mathtt{U}_\mathtt{A}\mathtt{D}_\mathtt{A}\mathtt{V}_\mathtt{A}^\top$: el coste del ajuste es el menor valor singular de $\mathtt{A}$ y el $\mathbf{g}$, que codifica la información de ω^* y $\mathtt{Q}_\infty^*$, es el vector singular derecho asociado (última columna de $\mathtt{V}_\mathtt{A}$).

Ya se conoce la matriz $\mathtt{G}$, ahora hay que hacer que se corresponda con el producto tensorial de $\mathbf{q}^*$ y $\boldsymbol{\omega}^\top$, ya que en la optimización lineal anterior no se ha impuesto que el vector solución $\mathbf{g}$ cumpla las restricciones propias de provenir de un producto tensorial como el considerado. Para extraer $\mathbf{q}^*$ y $\boldsymbol{\omega}^\top$ se toma la matriz de rango 1 que mejor aproxima la matriz $\mathtt{G}$ en norma inducida. Si no hubiese ruido, la matriz $\mathtt{G}$ sería de rango 1, por eso se toma esta aproximación. Una medida que indica la bondad del ajuste es el valor del segundo valor singular de $\mathtt{G}$, que como se ha dicho, teóricamente debería ser cero.

Esto se resuelve mediante la SVD de $\mathtt{G} = \mathtt{U}_\mathtt{G}\mathtt{D}_\mathtt{G}\mathtt{V}_\mathtt{G}^\top$. El vector de seis elementos que codifica la DIAC es la primera columna de la matriz $\mathtt{V}_\mathtt{G}$ (vector singular derecho asociado al mayor valor singular), mientras que el vector de 10 elementos que codifica la matriz $\mathtt{Q}_\infty^*$ es la primera columna de $\mathtt{U}_\mathtt{G}$ (vector singular izquierdo asociado al mayor valor singular).

El ultimo paso del algoritmo consiste en pasar de notación vectorial a notación matricial y mirar hasta qué punto se cumplen las restricciones propias del problema: la singularidad de la cuádrica dual

estimada. Una medida de la bondad del cumplimiento de la restricción es el menor valor singular de la cuádrica estimada.

El algoritmo no está muy bien explicado en el artículo de referencia y espero haber iluminado un poco más su desarrollo. Resumamos el citado algoritmo:

1. Construir la matriz de diseño $\mathtt{A}$ a partir de las matrices de proyección, siguiendo la estructura lineal de las ecuaciones en los elementos de $\mathbf{g} = \mathbf{q}^*\boldsymbol{\omega}^\top$
2. Hallar la solución SVD de $\mathtt{A} = \mathtt{U_A D_A V_A^\top}$. El vector $\mathbf{g}$ solución es la última columna de $\mathtt{V_A}$, el coste es el menor valor singular, $\sigma_{\mathtt{A}n} = \text{mín}\{\text{diag}(\mathtt{D_A}\})$.
3. Pasar de notación vectorial a notación matricial $\mathbf{g} \mapsto \mathtt{G}$.
4. Hallar la SVD de $\mathtt{G} = \mathtt{U_G D_G V_G^\top}$. La cuádrica estimada es la la primera columna de $\mathtt{U_G}$: $\mathbf{q}^* = \mathtt{U_G}(:, 1)$. La DIAC estimada es la primera columna de $\mathtt{V_G}$: $\boldsymbol{\omega} = \mathtt{V_G}(:, 1)$. El coste del ajuste en este paso en el segundo valor singular de $\sigma_{\mathtt{G}2}$.
5. Pasar de notación vectorial a ecuación matricial $\boldsymbol{\omega} \mapsto \omega^*$ y $\mathbf{q}^* \mapsto \mathtt{Q}^*$.
6. Analizar la cuádrica: el menor valor singular da una medida de la singularidad de la matriz. Proyectar la cuádrica sobre el espacio de las matrices de rango 3, anulando el último valor singular de $\mathtt{Q}^*$, para que sea $\mathtt{Q}^*_\infty$.
7. Calcular la matriz de parámetros intrínsecos $\mathtt{K}$ a partir de la DIAC, mediante la descomposición de Cholesky: $\omega^* = \mathtt{K}\mathtt{K}^\top$.
8. Calcular la homografía $\mathtt{H}$ que actualiza la reconstrucción a una de semejanza a partir de $\mathtt{Q}^*_\infty$. $\mathtt{H} = \mathtt{V}\boldsymbol{\Lambda}^{1/2}$, siendo $\mathtt{Q}^*_\infty \sim \mathtt{V}\boldsymbol{\Lambda}\mathtt{V}^\top$ la descomposición en autovalores y autovectores.

Algoritmo no lineal

El objetivo del algoritmo no lineal es minimizar la misma función de coste, pero imponiendo una restricción adicional:

$$\min_{\omega^*, \mathtt{Q}^*_\infty} \sum_{i=1}^{m} \|\epsilon_i\|^2 = \sum_{i=1}^{m} \|\omega^* \times (\mathtt{P}^i \mathtt{Q}^*_\infty \mathtt{P}^{i\top})\|^2 \quad \text{sujeto a} \quad \det \mathtt{Q}^*_\infty = 0$$

Este algoritmo también admite una función modelo, ya que el coste se puede expresar como la norma de un vector de error y el vector de medidas objetivo es el nulo.

$$f : (\omega^*, \mathtt{Q}^*) \equiv \mathbf{P} \longrightarrow \epsilon = \begin{bmatrix} \epsilon_1^\top & \cdots & \epsilon_i^\top & \cdots & \epsilon_m^\top \end{bmatrix}^\top$$

Se debe proporcionar a la función los datos auxiliares (las matrices de proyección $\mathtt{P}^i$) y la función de coste se puede calcular a partir de esta función modelo:

$$\text{coste} = \|\mathbf{0} - f(\mathbf{P})\|^2 = \|\epsilon\|^2 = \sum_{i=1}^{m} \|\epsilon_i\|^2$$

El algoritmo de optimización basado en la programación secuencial cuadrática (SQP) es el encargado de imponer las restricciones, pero necesita conocer la forma de éstas. Este algoritmo se explica en § 7.9.1.1. Además, durante la optimización se imponen dos restricciones sobre el factor de escala de la cónica dual y la cuádrica absoluta dual estimadas.

Para que los algoritmos de esta sección y la siguiente sean prácticos numéricamente es indispensable realizar una normalización afín de los datos antes de calcular las matrices de proyección $\mathtt{P}^i$ o aplicar una transformación similar a las matrices de proyección que ya se disponen para equilibrar las ecuaciones (7.13). La justificación se encuentra en el propio artículo [15].

7.6.2. Algoritmo 2

Hay una forma de construir la cuádrica absoluta dual, imponiendo la restricción de rango 3, si son conocidos el plano del infinito y la matriz de parámetros intrínsecos. Esto es posible siguiendo el resultado de § 7.2.3. Si la primera matriz de proyección es la canónica $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$, se utiliza la homografía de la ec. (7.5) para actualizar la reconstrucción. En una reconstrucción proyectiva, la cuádrica absoluta dual se puede descomponer de la forma $\mathtt{Q}^*_\infty = \mathtt{H}\,\mathrm{diag}(1,1,1,0)\mathtt{H}^\top$, (como se ha visto en la ec. (7.14)). Desarrollando esta expresión:

$$\mathtt{Q}^*_\infty = \mathtt{H}\,\mathrm{diag}(1,1,1,0)\mathtt{H}^\top = \begin{bmatrix} \mathtt{K}^1\mathtt{K}^{1\top} & -\mathtt{K}^1\mathtt{K}^{1\top}\mathbf{p} \\ -\mathbf{p}^\top\mathtt{K}^1\mathtt{K}^{1\top} & \mathbf{p}^\top\mathtt{K}^1\mathtt{K}^{1\top}\mathbf{p} \end{bmatrix} = \begin{bmatrix} \omega^{*1} & -\omega^{*1}\mathbf{p} \\ -\mathbf{p}^\top\omega^{*1} & \mathbf{p}^\top\omega^{*1}\mathbf{p} \end{bmatrix} \tag{7.15}$$

donde $\pi_\infty = (\mathbf{p}^\top, 1)^\top$.

Así que dados un plano y una matriz de parámetros intrínsecos, podemos calcular la supuesta DIAC $\omega^* = \mathtt{K}\mathtt{K}^\top$ y también la supuesta cuádrica absoluta dual (7.15). Si disponemos de una calibración proyectiva $\mathtt{P}^i$, podemos minimizar la misma función de coste que el algoritmo de Triggs, pero con esta nueva parametrización y sin necesidad de utilizar la SQP, sino técnicas estándar de optimización, como el algoritmo de Levenberg-Marquardt. Como estamos suponiendo cámaras iguales, $\mathtt{K}^1 = \mathtt{K}$.

$$\min_{\pi_\infty,\mathtt{K}} \sum_{i=1}^{m} \left\| \mathtt{K}\mathtt{K}^\top \times \left(\mathtt{P}^i \begin{bmatrix} \mathtt{K}\mathtt{K}^\top & -\mathtt{K}\mathtt{K}^\top\mathbf{p} \\ -\mathbf{p}^\top\mathtt{K}\mathtt{K}^\top & \mathbf{p}^\top\mathtt{K}\mathtt{K}^\top\mathbf{p} \end{bmatrix} \mathtt{P}^{i\top} \right) \right\|^2$$

El vector de parámetros tiene 3 coordenadas del plano del infinito y los 5 parámetros que definen la matriz de parámetros intrínsecos, $\mathtt{K}$, y como se ha dicho, los 8 parámetros pueden variar libremente y seguimos imponiendo la restricción de singularidad de la cuádrica absoluta dual. La función modelo tiene la misma salida que el algoritmo anterior, pero distinta parametrización:

$$f : (\pi, \mathtt{K}) \equiv \mathbf{P} \longrightarrow \epsilon = \begin{bmatrix} \epsilon_1^\top & \cdots & \epsilon_i^\top & \cdots & \epsilon_m^\top \end{bmatrix}^\top$$

Este método es muy parecido al que publicaron Heyden y Åström en [23].

7.6.3. Algoritmo 3

Una pequeña variación que se puede hacer sobre el anterior algoritmo es conservar el planteamiento, pero cambiar la función de coste: minimizar la resta unitaria de DIACs.

Definimos la distancia esférica de dos matrices $\mathtt{A}$ y $\mathtt{B}$ del mismo tamaño como:

$$d_e(\mathtt{A},\mathtt{B}) = \mathrm{mín}\left\{ \left| \frac{\mathtt{A}}{\|\mathtt{A}\|_F} - \frac{\mathtt{B}}{\|\mathtt{B}\|_F} \right|, \left| \frac{\mathtt{A}}{\|\mathtt{A}\|_F} + \frac{\mathtt{B}}{\|\mathtt{B}\|_F} \right| \right\} = \mathrm{mín}\left\{ \left| \mathcal{U}_F(\mathtt{A}) - \mathcal{U}_F(\mathtt{B}) \right|, \left| \mathcal{U}_F(\mathtt{A}) + \mathcal{U}_F(\mathtt{B}) \right| \right\}$$

donde el operador "unitarizar en norma Frobenius", $\mathcal{U}_F(\cdot)$, asigna a una matriz su matriz proporcional de norma unidad y orientada en el mismo sentido $\mathcal{U}_F(\mathtt{A}) = \mathtt{A}/\|\mathtt{A}\|_F$ (no $\mathcal{U}_F(\mathtt{A}) = -\mathtt{A}/\|\mathtt{A}\|_F$). En general, la distancia esférica se puede aplicar a cualquier par de vectores de la misma dimensión. Aquí se ha particularizado la definición para las matrices, cuyos elementos se pueden disponer en forma de vector.

Con esta definición, la función de coste se expresa:

$$\min_{\pi_\infty,\mathtt{K}} \sum_{i=1}^{m} d_e^2 \left(\mathtt{K}\mathtt{K}^\top, \mathtt{P}^i \begin{bmatrix} \mathtt{K}\mathtt{K}^\top & -\mathtt{K}\mathtt{K}^\top\mathbf{p} \\ -\mathbf{p}^\top\mathtt{K}\mathtt{K}^\top & \mathbf{p}^\top\mathtt{K}\mathtt{K}^\top\mathbf{p} \end{bmatrix} \mathtt{P}^{i\top} \right).$$

En el caso considerado, las matrices $\mathtt{A}$ y $\mathtt{B}$ son simétricas, luego se puede restringir la resta a los 6 elementos que forman la parte triangular superior, sin añadir redundancia.

La nueva función modelo tiene el mismo vector de parámetros (3 coordenadas de π_∞ y 5 parámetros de $\mathtt{K}$), mientras que el vector de medidas ya no es de dimensión $15m$, sino $9m$(resta de matrices 3×3) o $6m$ (resta de partes triangular superior). Este método es uno de los recogidos en [1].

Generalización. Los dos últimos algoritmos no lineales son inmediatamente generalizables al caso de cámaras con parámetros variables. Si $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$, entonces, las ecuaciones $\omega^{*i} \sim \mathtt{P}^i\mathtt{Q}^*_\infty\mathtt{P}^{i\top}$ se pueden escribir:

$$\mathtt{K}^i\mathtt{K}^{i\top} \sim \left(\mathtt{P}^i \begin{bmatrix} \mathtt{K}^1\mathtt{K}^{1\top} & -\mathtt{K}^1\mathtt{K}^{1\top}\mathbf{p} \\ -\mathbf{p}^\top\mathtt{K}^1\mathtt{K}^{1\top} & \mathbf{p}^\top\mathtt{K}^1\mathtt{K}^{1\top}\mathbf{p} \end{bmatrix} \mathtt{P}^{i\top}\right)$$

Se pueden minimizar las mismas funciones de coste (productos vectoriales de DIACs o restas unitarias de DIACs), pero cambiando el vector de parámetros sobre el que se optimiza, el cual incluye los, a lo sumo, $5m$ parámetros de las $\mathtt{K}^i$ más los tres parámetros del plano del infinito ($\mathbf{p}$). El vector de parámetros se puede reducir si se conocen restricciones internas de las cámaras.

7.7. Algoritmo de Hartley

El algoritmo de este apartado se explica en [6]. Pongamos la función de coste a minimizar y a continuación expliquemos de dónde se deduce:

$$\sum_{i=1}^{m} \|\alpha_i X_i - \mathtt{I}\|^2 \tag{7.16}$$

Utilizando una vez más el resultado de § 7.2.3. Si la primera matriz de proyección es la canónica $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$, se utiliza la homografía de la ec. (7.5) para actualizar la reconstrucción. Supongamos que las matrices de proyección en la referencia proyectiva se expresan: $\mathtt{P}^i = [\mathtt{A}^i \mid -\mathtt{A}^i\mathbf{t}^i]$ y queremos llegar a las matrices métricas $\mathtt{P}^i_{\mathrm{M}} = \mathtt{P}^i\mathtt{H} = \mathtt{K}[\mathtt{R}^i \mid -\mathtt{R}^i\widetilde{\mathbf{C}}^i]$. Realicemos el producto $\mathtt{P}^i\mathtt{H}$:

$$\mathtt{P}^i\mathtt{H} = [\mathtt{A}^i \mid -\mathtt{A}^i\mathbf{t}^i] \begin{bmatrix} \mathtt{K} & \mathbf{0} \\ -\mathbf{p}^\top\mathtt{K} & 1 \end{bmatrix} = [\mathtt{A}^i\mathtt{K} + \mathtt{A}^i\mathbf{t}^i\mathbf{p}^\top\mathtt{K} \mid -\mathtt{A}^i\mathbf{t}^i] = [\mathtt{A}^i(\mathtt{I} + \mathbf{t}^i\mathbf{p}^\top)\mathtt{K} \mid -\mathtt{A}^i\mathbf{t}^i]$$

e identifiquemos elementos con $\mathtt{P}^i_{\mathrm{M}}$. De la última columna de las matrices no se puede extraer información alguna en un primer momento, en cambio, la submatriz formada por las tres primeras columnas tiene restricciones que se pueden (y deben) explotar.

$$\mathtt{A}^i(\mathtt{I} + \mathbf{t}^i\mathbf{p}^\top)\mathtt{K} \approx \mathtt{K}\mathtt{R}^i \tag{7.17}$$

Ya falta menos: tomemos la descomposición QR de la submatriz, $\mathtt{A}^i(\mathtt{I}+\mathbf{t}^i\mathbf{p}^\top)\mathtt{K} \sim \mathtt{K}^{i'}\mathtt{R}^i$. A continuación se calculan las matrices $X_i = \mathtt{K}^{-1}\mathtt{K}^{i'}$, $\forall i$. Para la primera matriz de proyección es obvio que $X_1 = \mathtt{I}$, para las demás cámaras, si los $\mathtt{K}$ y $\mathbf{p}$ candidatos cumplen la aproximación (7.17), entonces se verificará $X_i \approx \mathtt{I}$. Así que una función de coste puede indicar cuánto se alejan las matrices X_i respecto de la identidad y esa es (7.16).

Como se trabaja en coordenadas homogéneas, se incluye el factor de proporcionalidad α_i. El subíndice $i = 1, \ldots, m$ podría empezar en la cámara 2, pero se deja así porque en los anteriores algoritmos tampoco se ha cambiado. La hipótesis de que la primera matriz de proyección es la canónica y la homografía $\mathtt{H}$ es (7.5) implica que suponemos que la primera cámara no tiene errores: de ahí el sesgo de estos algoritmos porque no tratan por igual a todas las cámaras. A mi juicio parece más cómodo minimizar la distancia esférica (resta unitaria) de matrices, ya que así se evita seguir la pista de los factores de proporcionalidad α_i.

$$\min_{\mathbf{p},\mathtt{K}} \sum_{i=1}^{m} d_e^2(X_i, \mathtt{I}) \tag{7.18}$$

7.8. Horópteras

Esta sección está dedicada a los algoritmos de autocalibración basados en la curva horóptera que se define entre cada par de cámaras. La referencia básica es [9].

La sección se estructura de la siguiente manera: primero se define la horóptera asociada a un par de cameras, después se citan sus propiedades más relevantes, a continuación se contempla la forma de parametrizar una horóptera y de relacionarla con las matrices de proyección; se muestra qué efecto tiene una homografía del espacio sobre una horóptera y por último se explican tres algoritmos de estimación de objetos de autocalibración para recuperar la estructura euclídea del espacio.

Definición

Dado un par de matrices de proyección, podemos calcular el conjunto de puntos del espacio que se ven igual desde las dos cámaras (se proyectan en puntos de iguales coordenadas en ambas imágenes $\mathbf{x} = \mathbf{x}'$). Este conjunto de puntos es una *horóptera*: una curva cúbica del espacio, llamada cúbica alabeada (twisted cubic).

7.8.1. Propiedades de la horóptera

Presentamos varias de las propiedades de la horóptera sin demostrarlas; muchas de ellas no servirán para interpretar la geometría de la disposición de cámaras y para recuperar la estructura euclídea del espacio mediante la estimación de los objetos de autocalibración.

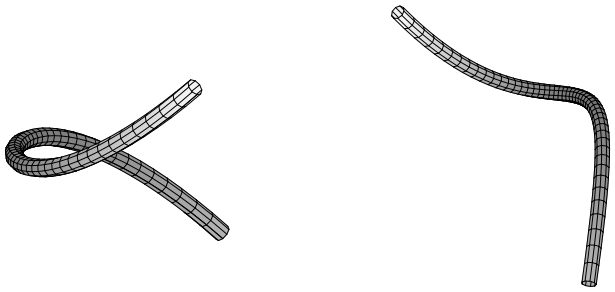


Figura 7.4: Cúbica alabeada

- La horóptera pasa por los centros de proyección de las dos cámaras que la definen.
- Debido a la propiedad anterior, la proyección de la horóptera en el plano de la imagen es una cónica: disminuye un grado (de 3 a 2). Dicha cónica es la parte simétrica de la matriz fundamental entre ambas cámaras: $\mathtt{F}_\mathtt{S} = (\mathtt{F} + \mathtt{F}^\top)/2$.
- Cualquier horóptera corta a un plano en tres puntos, por ser una cúbica. Los puntos de corte provienen de la solución de una ecuación de tercer grado.
- La horóptera corta al plano del infinito π_∞ en un punto real y dos complejos conjugados, de forma que:
  - los puntos complejos conjugados están sobre la cónica absoluta Ω_∞ y el primero es el polo respecto de ésta de la recta que une a los anteriores.
  - El punto real es la dirección del eje de rotación del movimiento entre las dos cámaras.

  - La recta que une los puntos complejos conjugados (la polar del punto real) da la dirección del haz de planos perpendiculares al eje de rotación.
  - Los parámetros que aplicados a la parametrización de la horóptera definen estos puntos tienen el mismo módulo.

- La horóptera es asintótica al eje del movimiento y está sobre un cilindro de base circular.
- El teorema de Steiner se aplica a la twisted cubic. Es una forma de parametrizar la horóptera.

  Supongamos la situación en la que tenemos dos haces de rectas en $\mathbb{P}^3$ y establecemos una homografía que relaciona los dos haces: unas rectas con sus transformadas por la homografía. En general, una recta y su transformada no se cortarán, sino que se cruzarán. Pero hay pares de rectas correspondientes que sí se cortan en un punto. Dicho punto está sobre la cúbica alabeada (horóptera).

  Si los haces son los de los rayos que salen de los centros ópticos de las dos cámaras y la homografía es la inducida por el movimiento de la cámara, entonces hallamos la horóptera de nuestra aplicación.

- Casos degenerados
  - En un movimiento de traslación y rotación de 0 ó π, la horóptera degenera en el plano del infinito: los puntos que se ven igual desde las dos cámaras son, precisamente, los del plano del infinito.
  - Cuando consideramos cámaras "planas" (su movimiento es el que tendría si se moviera sobre una mesa), la horóptera degenera en una recta y una circunferencia. La recta es la definida por el eje de rotación del movimiento entre las dos cámaras. La circunferencia considerada como completa, pasa por los centros ópticos de las cámaras y está relacionada con el arco capaz del segmento que une los centros ópticos.

7.8.2. Relación de la horóptera con las matrices de proyección

La representación paramétrica que se ha elegido para una horóptera es el siguiente producto matricial:

$$\begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix} = \mathtt{H} \begin{pmatrix} 1 \\ \theta \\ \theta^2 \\ \theta^3 \end{pmatrix} = \begin{pmatrix} H_{11} + H_{12}\theta + H_{13}\theta^2 + H_{14}\theta^3 \\ H_{21} + H_{22}\theta + H_{23}\theta^2 + H_{24}\theta^3 \\ H_{31} + H_{32}\theta + H_{33}\theta^2 + H_{34}\theta^3 \\ H_{41} + H_{42}\theta + H_{43}\theta^2 + H_{44}\theta^3 \end{pmatrix} \tag{7.19}$$

Lo podemos resumir en

$$\mathbf{h}(\theta) = \mathtt{H}\Theta \tag{7.20}$$

La fórmula que calcula la matriz de la horóptera $\mathtt{H}$ asociada a un par de proyecciones es.

$$\mathbf{h}(\theta) = \ker(\mathtt{P}_1 - \theta\mathtt{P}_2) \tag{7.21}$$

7.8.3. Transformación proyectiva de una horóptera y corte con π_∞

Los puntos de corte de la horóptera con el plano del infinito vienen determinados por los valores del parámetro θ solución de la ecuación:

$$\pi_\infty^\top \mathbf{h}(\theta) = 0$$

Si ahora se aplica una homografía arbitraria $\mathtt{G}$ a los datos (matrices de proyección euclídeas), se obtienen nuevas matrices de proyección y otra expresión de la horóptera:

$$\tilde{\mathbf{h}}(\theta) = \ker(\mathtt{P}_1\mathtt{G} - \theta\mathtt{P}_2\mathtt{G}) = \ker((\mathtt{P}_1 - \theta\mathtt{P}_2)\mathtt{G}) = \mathtt{G}^{-1}\mathbf{h}(\theta)$$

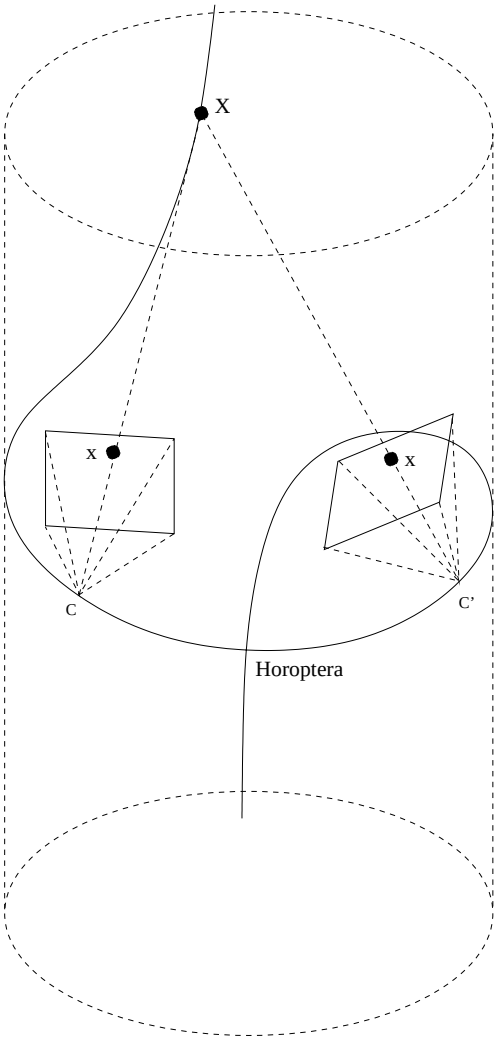


Figura 7.5: La horóptera asociada a un par de cámaras

La última igualdad se deduce de la equivalencia

$$(\mathtt{P}_1 - \theta\mathtt{P}_2)\mathbf{h}(\theta) = 0 \Leftrightarrow (\mathtt{P}_1 - \theta\mathtt{P}_2)\mathtt{G}\ \mathtt{G}^{-1}\mathbf{h}(\theta) = 0$$

Sustituimos la nueva expresión de la horóptera, $\mathbf{h}(\theta) = \mathtt{G}\tilde{\mathbf{h}}(\theta)$, en la ecuación de incidencia con el plano del infinito:

$$\pi_\infty^\top \mathbf{h}(\theta) = \pi_\infty^\top \mathtt{G}\tilde{\mathbf{h}}(\theta) = \tilde{\pi}_\infty^\top \tilde{\mathbf{h}}(\theta) = 0$$

donde se deduce que

$$\tilde{\pi}_\infty^\top = \pi_\infty^\top \mathtt{G} \qquad \tilde{\pi}_\infty = \mathtt{G}^\top \pi_\infty$$

En el marco euclídeo, $\pi_\infty = (0,0,0,1)^\top$, así que $\tilde{\pi}_\infty$ es la cuarta fila de la matriz $\mathtt{G}$, la homografía que transforma lo euclídeo en lo proyectivo.

También se podía haber razonado de otro modo, mirando la condición de pertenencia de un punto al plano del infinito en las dos referencias. Los puntos 3D se transforman al contrario que las matrices de proyección, como se indica en § 6.6.3. Así que

$$\pi_\infty^\top \mathbf{X} = \pi_\infty^\top \mathtt{G}\ \mathtt{G}^{-1}\mathbf{X} = \tilde{\pi}_\infty^\top \tilde{\mathbf{X}} = 0$$

7.8.4. Algoritmo 1: horópteras y la Restricción Unimodular

En los algoritmos que utilizan horópteras suponemos que las cámaras son idénticas: mismos parámetros intrínsecos `K` y distintas rotaciones y traslaciones. Partimos de una calibración proyectiva de las cámaras y queremos llegar a una calibración euclídea.

El objetivo de esta sección es discutir el primer algoritmo propuesto en [9], que es la versión de la restricción unimodular expresada mediante la propiedad de los parámetros de los puntos de corte de las horópteras entre cada par de cámaras con el plano del infinito: todos estos parámetros tienen el mismo módulo.

7.8.4.1. Algoritmo completo

Este algoritmo es el primero propuesto en [9]. Incluye 6 términos de coste por cada terna de puntos de corte de un plano de prueba con cada horóptera.

Para un plano cualquiera, hallamos los tres instantes de corte con cada horóptera k-ésima: $\{\theta_1^k, \theta_2^k, \theta_3^k\}$. Estos tres instantes de corte se obtienen de resolver una ecuación de tercer grado, que puede tener tres soluciones reales o una solución real y dos complejas conjugadas. Ordenemos las raíces para que θ_1^k siempre sea una raíz real y $\{\theta_2^k, \theta_3^k\}$ sean las otras raíces, normalmente complejas conjugadas. En caso de tres raíces reales, se ordenan en orden creciente.

Para n cámaras, hay $NH = \binom{n}{2} = n(n-1)/2$ horópteras entre pares de cámaras, lo que proporciona $\binom{n}{2} = n(n-1)/2$ ternas de puntos de corte.

Función de coste

El plano del infinito se obtiene optimizando una función de coste que asigna a un **plano** candidato el valor:

$$c_1(\mathbf{plano}) = \sum_{k=1}^{NH} \sum_{\substack{m,n=1 \\ m \neq n}}^{3} \left(\left| \frac{\theta_m^k}{\theta_n^k} \right| - 1 \right)^2 \tag{7.22}$$

Si se expande la expresión anterior para mayor claridad, se obtienen 6 términos para cada k:

$$\sum_{k=1}^{NH} \left(\left| \frac{\theta_1^k}{\theta_2^k} \right| - 1 \right)^2 + \left(\left| \frac{\theta_1^k}{\theta_3^k} \right| - 1 \right)^2 + \left(\left| \frac{\theta_2^k}{\theta_1^k} \right| - 1 \right)^2 + \left(\left| \frac{\theta_3^k}{\theta_1^k} \right| - 1 \right)^2 + \left(\left| \frac{\theta_2^k}{\theta_3^k} \right| - 1 \right)^2 + \left(\left| \frac{\theta_3^k}{\theta_2^k} \right| - 1 \right)^2 \tag{7.23}$$

Función modelo

Esta función de coste se puede expresar como norma de un vector de error, así que admite una función modelo, la cual permite aplicar el algoritmo de Levenberg-Marquardt y así facilitar la optimización.

$$f : \mathbf{plano} \equiv \mathbf{P} \rightarrow \widehat{\mathbf{X}} \equiv \begin{bmatrix} \widehat{\mathbf{X}}_1 \\ \vdots \\ \widehat{\mathbf{X}}_k \\ \vdots \\ \widehat{\mathbf{X}}_{NH} \end{bmatrix}$$

con

$$\widehat{\mathbf{X}}_k = \left[\left| \frac{\theta_1^k}{\theta_2^k} \right| - 1, \left| \frac{\theta_1^k}{\theta_3^k} \right| - 1, \left| \frac{\theta_2^k}{\theta_1^k} \right| - 1, \left| \frac{\theta_3^k}{\theta_1^k} \right| - 1, \left| \frac{\theta_2^k}{\theta_3^k} \right| - 1, \left| \frac{\theta_3^k}{\theta_2^k} \right| - 1 \right]^\top \tag{7.24}$$

El vector de parámetros $\mathbf{P}$ contiene una parametrización de un plano de $\mathbb{P}^3$, a partir del cual se calculan los $3NH$ puntos de corte con las NH horópteras y sus parámetros, θ_i^k, $i = 1 \ldots 3$. Con estos

parámetros se construye el vector de medidas $\widehat{\mathbf{X}}$ de dimensión $6NH$, que coincide con el vector de error ϵ, puesto que el vector de medidas objetivo es el vector nulo. Al definir $\widehat{\mathbf{X}}$ de esta forma, su norma al cuadrado es el coste (7.22).

$$\text{coste} = \|\epsilon\|^2 = \|\mathbf{X} - \widehat{\mathbf{X}}\|^2 = \|\mathbf{0} - f(\mathbf{P})\|^2 = \|f(\mathbf{P})\|^2 = \sum_{k=1}^{NH} \|\widehat{\mathbf{X}}_k\|^2 = \sum_{k=1}^{NH} \sum_{\substack{m,n=1 \\ m\neq n}}^{3} \left(\left| \frac{\theta_m^k}{\theta_n^k} \right| - 1 \right)^2$$

Parametrizaciones de un plano de $\mathbb{P}^3$

Además de las funciones modelo y de coste, podemos pensar en tres parametrizaciones del plano de prueba, por lo que en total hay 6 formas distintas de calcular el mismo coste. Las parametrizaciones son las siguientes:

- Coordenadas homogéneas. **plano** $= \mathbf{u} = (u_1, u_2, u_3, u_4)^\top$, salvo escalado: $\|\mathbf{u}\| = 1$.
- Coordenadas esféricas del plano, es decir, expresar las coordenadas del plano como un punto sobre la esfera $\mathbb{S}^3$, que es una doble cubierta de $\mathbb{P}^3$. **plano** $= \boldsymbol{\varphi} = (\varphi_1, \varphi_2, \varphi_3)^\top$, con $0 \leq \varphi_1 \leq 2\pi$, $0 \leq \varphi_2 \leq \pi$, $0 \leq \varphi_3 \leq \pi$.
- Instantes de corte reales del plano con 3 horópteras (habitualmente, las tres primeras). **plano** $= \mathbf{t} = (t_1, t_2, t_3)^\top = (\theta_1^1, \theta_1^2, \theta_1^3)^\top$.

Las dos últimas son parametrizaciones mínimas, es decir, que emplean el mínimo número de parámetros (tres números), mientras que la primera es una sobreparametrización porque utiliza 4 números, salvo un grado de libertad por el escalado.

7.8.4.2. Algoritmo simplificado

Este algoritmo es una simplificación del anterior porque sólo incluye 2 de los 6 términos de coste de (7.22) por cada horóptera.

Supongamos que los instantes de corte de las horópteras $\{\theta_1^k, \theta_2^k, \theta_3^k\}$ son raíces tales que θ_1^k siempre es real y $\{\theta_2^k, \theta_3^k\}$ son complejas conjugadas: nunca se da el caso de tres raíces reales. Bajo estas suposiciones, los dos últimos términos de coste de (7.23) son siempre cero. Este algoritmo no penaliza el caso de tres instantes de corte reales entre un plano y una horóptera, sin embargo no es una situación habitual, ni siquiera si se inicializa suponiendo cámaras ortogonales. Además, se ha comprobado que numéricamente basta con minimizar utilizando los dos primeros términos del coste de (7.23).

Función de coste

La función de coste (7.22) queda simplificada:

$$\tilde{c}_1(\textbf{plano}) = \sum_{k=1}^{NH} \left(\left| \frac{\theta_1^k}{\theta_2^k} \right| - 1 \right)^2 + \left(\left| \frac{\theta_1^k}{\theta_3^k} \right| - 1 \right)^2 \tag{7.25}$$

Función modelo

La función modelo también se simplifica. Lo único que hay que cambiar es el vector $\widehat{\mathbf{X}}_k$ con los términos de coste de cada horóptera.

$$\widehat{\mathbf{X}}_k = \left[\left| \frac{\theta_1^k}{\theta_2^k} \right| - 1, \left| \frac{\theta_1^k}{\theta_3^k} \right| - 1 \right]^\top$$

El vector de medidas $\widehat{\mathbf{X}}$ tiene dimensión $2NH$. La norma al cuadrado del vector $\widehat{\mathbf{X}}$ es el coste (7.25).

$$\text{coste} = \|\epsilon\|^2 = \|\mathbf{X} - \widehat{\mathbf{X}}\|^2 = \|\mathbf{0} - f(\mathbf{P})\|^2 = \|f(\mathbf{P})\|^2 = \sum_{k=1}^{NH} \|\widehat{\mathbf{X}}_k\|^2 = \sum_{k=1}^{NH} \left(\left| \frac{\theta_1^k}{\theta_2^k} \right| - 1 \right)^2 + \left(\left| \frac{\theta_1^k}{\theta_3^k} \right| - 1 \right)^2$$

Este algoritmo también admite las tres parametrizaciones dadas en § 7.8.4.1 para el plano de prueba (candidato a plano del infinito).

7.8.5. Algoritmo 2: horópteras y la Cónica Absoluta

El segundo algoritmo es el segundo propuesto en [9] y consiste en utilizar las propiedades de los puntos de corte de las horópteras entre cada par de cámaras con el plano del infinito, para hallar éste y la cónica absoluta. Se necesitan como mínimo 3 cámaras (2 horópteras).

Para un plano cualquiera y n cámaras hay $NH = n(n-1)/2$ ternas de puntos de corte con las horópteras. Con todos estos puntos y las propiedades de polaridad y pertenencia a la cónica absoluta, queremos estimar la cónica que mejor se ajusta. Diremos que esta cónica es la candidata a ser cónica absoluta Ω_∞. Además, se deben verificar otras restricciones: la matriz de Ω_∞ debe tener todos sus autovalores del mismo signo (positivos), para que sea semejante a la matriz identidad.

Según se explica en [9], en lugar de estimar la cónica absoluta, proyectaremos las ternas de puntos de corte sobre el plano virtual de la retina a través de las n matrices de proyección (porque todas las cámaras comparten la misma matriz de parámetros intrínsecos). Con todos estos puntos de corte, se estima la IAC o PAC que mejor se ajusta a las propiedades de pertenencia y polaridad, puesto que éstas son preservadas por la proyección. Una característica relevante de la geometría del problema es que la cónica absoluta Ω_∞ se proyecta sobre la misma cónica ω independientemente de la matriz de proyección. Hay una sola IAC, puesto que hay una sola matriz de parámetros intrínsecos.

El problema es: dadas varias ternas de puntos en un plano (uno real y dos complejos conjugados), proyectar los puntos sobre el plano virtual de la retina y calcular la cónica que mejor se ajusta según las propiedades que debe verificar cada terna (dos de polaridad y dos de pertenencia). La cónica estimada es la candidata a IAC.

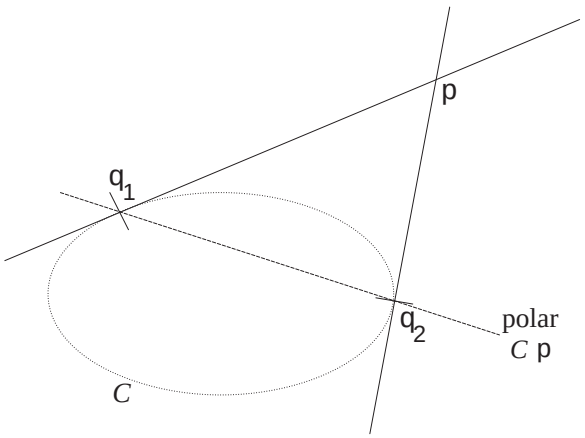


Figura 7.6: Configuración básica de una terna de puntos de corte, respecto de la cónica a estimar

El ajuste de la cónica se realiza mediante un algoritmo lineal y se puede hacer tanto en su versión de autovalores como en la de valores singulares.

7.8.5.1. Explicación del ajuste

En $\mathbb{P}^2$, una cónica está representada por una matriz C simétrica y de 3×3.

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & c_{23} \\ c_{13} & c_{23} & c_{33} \end{bmatrix}$$

La ecuación que verifica un punto $\mathbf{q}$ que pertenece a la cónica es $\mathbf{q}^\top C\mathbf{q} = 0$. Las coordenadas de la polar de un punto $\mathbf{p}$ son $\mathbf{l} = C\mathbf{p}$, es decir, $\mathbf{l}^\top = \mathbf{p}^\top C^\top = \mathbf{p}^\top C$ y la condición que verifica un punto $\mathbf{q}$ sobre dicha recta $\mathbf{l}$ es $\mathbf{l}^\top\mathbf{q} = \mathbf{p}^\top C\mathbf{q} = 0$.

Dados los 3 puntos de la terna, se deben verificar 4 ecuaciones lineales en los elementos de la cónica.

$$(\mathbf{p}, \mathbf{q}_1, \mathbf{q}_2) \longrightarrow \begin{cases} \mathbf{q}_1^\top C\mathbf{q}_1 = 0 \\ \mathbf{q}_2^\top C\mathbf{q}_2 = 0 \\ \mathbf{p}^\top C\mathbf{q}_1 = 0 \\ \mathbf{p}^\top C\mathbf{q}_2 = 0 \end{cases}$$

Las dos primeras reflejan la pertenencia de los puntos complejos conjugados a la cónica absoluta Ω_∞ y las dos segundas representan la relación de polaridad entre el punto real y los dos complejos conjugados, respecto de la misma cónica.

Además, como venimos diciendo, $\mathbf{q}_2 = \mathbf{q}_1^*$.

La relación $\mathbf{x}^\top C\mathbf{y} = 0$ se puede poner en forma de producto matricial de vectores. Sean $\mathbf{x} = (x_1, x_2, x_3)^\top$, $\mathbf{y} = (y_1, y_2, y_3)^\top$ y llamemos

$$\mu_{\mathbf{xy}} \equiv \mu(\mathbf{x}, \mathbf{y}) = [x_1y_1 \quad x_1y_2 + x_2y_1 \quad x_1y_3 + x_3y_1 \quad x_2y_2 \quad x_2y_3 + x_3y_2 \quad x_3y_3]^\top$$

Además, se verifica

$$\mu(\mathbf{x}^*, \mathbf{y}^*) = \mu^*(\mathbf{x}, \mathbf{y}) \equiv \mu^*_{\mathbf{xy}} \tag{7.26}$$

Entonces, las condiciones se traducen en:

$$\begin{aligned} \mathbf{q}_i^\top C\mathbf{q}_i &= \mu_{\mathbf{q}_i\,\mathbf{q}_i}^\top \mathbf{c} = 0 \qquad i = 1 \ldots 2 \\ \mathbf{p}^\top C\mathbf{q}_i &= \mu_{\mathbf{p}\,\mathbf{q}_i}^\top \mathbf{c} = 0 \qquad i = 1 \ldots 2 \end{aligned}$$

$\mathbf{c} = [c_{11} \quad c_{12} \quad c_{13} \quad c_{22} \quad c_{23} \quad c_{33}]^\top$ es la cónica, representada por un vector de 6×1.

7.8.5.2. Enfoque basado en autovalores

En el caso de datos ruidosos, las condiciones anteriores no se verificarán de forma exacta y se puede tratar de encontrar la cónica del siguiente problema de minimización:

$$\min_C \sum_{k=1}^{NH} \sum_{i=1}^{2} \left(|\mathbf{q}_{ik}^\top C\mathbf{q}_{ik}|^2 + |\mathbf{p}_k^\top C\mathbf{q}_{ik}|^2 \right) = \min_{\|\mathbf{c}\|=1} \sum_{k=1}^{NH} \sum_{i=1}^{2} \left(|\mu_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}}^\top \mathbf{c}|^2 + |\mu_{\mathbf{p}_k\,\mathbf{q}_{ik}}^\top \mathbf{c}|^2 \right) \tag{7.27}$$

siendo NH el número de horópteras.

Podemos desarrollar la expresión anterior para llegar a la conclusión de que queremos minimizar un polinomio cuadrático en los elementos de $\mathbf{c}$ sujeto a la restricción $\|\mathbf{c}\| = 1$. Dicho problema lo podemos expresar como

$$\min_{\|\mathbf{c}\|=1} g(\mathbf{c}) = \min_{\|\mathbf{c}\|=1} \mathbf{c}^\top \mathtt{M}\mathbf{c}$$

siendo $\mathtt{M}$ una matriz simétrica de 6×6. Para hallar dónde se alcanza el mínimo hay que utilizar los multiplicadores de Lagrange. La función de Lagrange es:

$$F(\mathbf{c}, z) = g(\mathbf{c}) + z(\|\mathbf{c}\|^2 - 1) = \mathbf{c}^\top \mathtt{M}\mathbf{c} + z(\mathbf{c}^\top \mathbf{c} - 1)$$

Si se iguala a cero su gradiente (respecto a $\mathbf{c}$ y a z) se obtiene un sistema de ecuaciones que indica los puntos críticos candidatos a minimizar g sujeta a la restricción $\|\mathbf{c}\|^2 = 1$.

$$\mathbf{0} = \nabla F = \begin{cases} \dfrac{\partial F}{\partial \mathbf{c}} &= \dfrac{\partial}{\partial \mathbf{c}}\left(\mathbf{c}^\top \mathtt{M}\mathbf{c}\right) + z\dfrac{\partial}{\partial \mathbf{c}}\left(\mathbf{c}^\top \mathbf{c} - 1\right) = 2\mathtt{M}\mathbf{c} + 2z\mathbf{c} \\ \dfrac{\partial F}{\partial z} &= \mathbf{c}^\top \mathbf{c} - 1 \end{cases}$$

De la primera ecuación resulta el sistema

$$\mathtt{M}\mathbf{c} = -z\mathbf{c} \tag{7.28}$$

de donde se deduce que $\mathbf{c}$ es un autovector de $\mathtt{M}$ con autovalor $\lambda = -z$. ¿Qué autovalor escoger? El que minimiza el coste, que utilizando la segunda ecuación, es:

$$\mathbf{c}^\top \mathtt{M}\mathbf{c} = \mathbf{c}^\top(-z\mathbf{c}) = \lambda \mathbf{c}^\top \mathbf{c} = \lambda$$

Como el coste el autovalor, se elige el menor de todos. Así pues, la solución es el autovector de $\mathtt{M}$ correspondiente al menor autovalor. El coste del ajuste es el menor autovalor de $\mathtt{M}$, como se ha demostrado.

Al ser $\mathtt{M}$ una matriz simétrica, es no definida negativa, por lo que $g(\mathbf{c}) \geq 0 \;\; \forall \mathbf{c} \neq \mathbf{0}$.

DEMO

Desarrollemos la expresión (7.27) y tengamos en cuenta que buscamos una cónica de coeficientes reales, por lo que $\mathbf{c}^* = \mathbf{c}$.

$$\begin{aligned}
&\sum_{k=1}^{NH}\sum_{i=1}^{2}\left(|\mu^\top_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}}\mathbf{c}|^2 + |\mu^\top_{\mathbf{p}_k\,\mathbf{q}_{ik}}\mathbf{c}|^2\right) &= \\
&\sum_{k=1}^{NH}\sum_{i=1}^{2}\left((\mu^\top_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}}\mathbf{c})(\mu^{*\top}_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}}\mathbf{c}) + (\mu^\top_{\mathbf{p}_k\,\mathbf{q}_{ik}}\mathbf{c})(\mu^{*\top}_{\mathbf{p}_k\,\mathbf{q}_{ik}}\mathbf{c})\right) &= \\
&\sum_{k=1}^{NH}\sum_{i=1}^{2}\left((\mathbf{c}^\top\mu_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}})(\mu^{*\top}_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}}\mathbf{c}) + (\mathbf{c}^\top\mu_{\mathbf{p}_k\,\mathbf{q}_{ik}})(\mu^{*\top}_{\mathbf{p}_k\,\mathbf{q}_{ik}}\mathbf{c})\right) &= \\
&\sum_{k=1}^{NH}\sum_{i=1}^{2}\left(\mathbf{c}^\top \mathtt{M}_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}}\mathbf{c} + \mathbf{c}^\top \mathtt{M}_{\mathbf{p}_k\,\mathbf{q}_{ik}}\mathbf{c}\right) &= \\
&\mathbf{c}^\top\left(\sum_{k=1}^{NH}\sum_{i=1}^{2}\left(\mathtt{M}_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}} + \mathtt{M}_{\mathbf{p}_k\,\mathbf{q}_{ik}}\right)\right)\mathbf{c} &= \\
&\mathbf{c}^\top\left(\sum_{k=1}^{NH}\mathtt{M}_k\right)\mathbf{c} &= \\
&\mathbf{c}^\top\mathtt{M}\mathbf{c}
\end{aligned}$$

Hemos llamado

$$\mu_{\mathbf{p}_k\,\mathbf{q}_{ik}}\,\mu^{*\top}_{\mathbf{p}_k\,\mathbf{q}_{ik}} = \mathtt{M}_{\mathbf{p}_k\,\mathbf{q}_{ik}}$$

$$\mu_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}}\,\mu^{*\top}_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}} = \mathtt{M}_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}}$$

Además, se verifica que

$$\begin{aligned}\mathtt{M}_{\mathbf{p}_k\,\mathbf{q}_{2k}} &= \mathtt{M}^{*}_{\mathbf{p}_k\,\mathbf{q}_{1k}}\\ \mathtt{M}_{\mathbf{q}_{2k}\,\mathbf{q}_{2k}} &= \mathtt{M}^{*}_{\mathbf{q}_{1k}\,\mathbf{q}_{1k}}\end{aligned}$$

por lo que

$$\mathtt{M}_k = \sum_{i=1}^{2}\left(\mathtt{M}_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}} + \mathtt{M}_{\mathbf{p}_k\,\mathbf{q}_{ik}}\right) = 2\,\mathrm{Re}\left[\mathtt{M}_{\mathbf{q}_{1k}\,\mathbf{q}_{1k}} + \mathtt{M}_{\mathbf{p}_k\,\mathbf{q}_{1k}}\right]$$

y la matriz $\mathtt{M}$ queda al final:

$$\mathtt{M} = \sum_{k=1}^{NH}\mathtt{M}_k = \sum_{k=1}^{NH} 2\,\mathrm{Re}\left[\mathtt{M}_{\mathbf{q}_{1k}\,\mathbf{q}_{1k}} + \mathtt{M}_{\mathbf{p}_k\,\mathbf{q}_{1k}}\right] \tag{7.29}$$

Interpretación geométrica

Se puede ver $\mathtt{M}$ como la matriz de una cuádrica en $\mathbb{P}^5$. Si esta cuádrica es degenerada, el coste siempre es cero. Con los datos exactos, $\mathtt{M}$ es *semidefinida positiva*: tiene un autovalor nulo y el resto positivos. Así que como cuádrica real es un único punto (sólo hay una cónica real que se ajuste a los datos). El resto de puntos de la cuádrica son complejos. Con datos ruidosos, $\mathtt{M}$ sigue siendo simétrica, pero no tiene por qué tener un autovalor nulo. En principio será *definida positiva* (todos los autovalores serán positivos) y, como ya se ha dicho, se toma como coste del ajuste el menor autovalor.

7.8.5.3. Enfoque SVD

Otro enfoque equivalente consiste en poner las condiciones que deben verificarse una a continuación de otra en forma matricial. Si lo hacemos para todas las ternas de puntos, obtendremos la matriz de diseño $\mathtt{A}$, de tamaño $4NH \times 6$, del sistema homogéneo $\mathtt{A}\mathbf{c} = \mathbf{0}$.

$$\mathtt{A}_k = \begin{bmatrix} \mu^{\top}_{\mathbf{q}_{1k}\,\mathbf{q}_{1k}}\\ \mu^{\top}_{\mathbf{q}_{2k}\,\mathbf{q}_{2k}}\\ \mu^{\top}_{\mathbf{p}_k\,\mathbf{q}_{1k}}\\ \mu^{\top}_{\mathbf{p}_k\,\mathbf{q}_{2k}}\end{bmatrix} \qquad \mathtt{A} = \begin{bmatrix}\mathtt{A}_1\\ \mathtt{A}_2\\ \vdots\\ \mathtt{A}_{NH}\end{bmatrix}$$

En el caso de datos exactos ruidosos, no se verificará la anterior ecuación matricial, sino que habrá un vector de error o residuo $\mathtt{A}\mathbf{c} = \epsilon$, y sería deseable minimizar la norma de dicho vector de error:

$$\text{mín}\,\|\epsilon\|^2 = \min_{\|\mathbf{c}\|=1}\|\mathtt{A}\mathbf{c}\|^2$$

El problema y su solución son clásicos, similares a los comentados para la matriz fundamental en § 6.2.3.2. La solución se obtiene mediante la SVD de $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^{\top}$. La cónica $\mathbf{c}$ es la columna de $\mathtt{V}$ correspondiente al menor valor singular de $\mathtt{A}$, es decir, la sexta columna. En la aplicación concreta, la matriz $\mathtt{A}$ es compleja con filas dos a dos conjugadas. La matriz $\mathtt{V}$ es real (la matriz $\mathtt{U}$, compleja), por lo que $\mathbf{c}$ es real y la matriz C también.

Todavía se puede simplificar más la matriz $\mathtt{A}$ explotando la propiedad de conjugación de las filas, y así obtener una matriz equivalente, pero de la mitad de filas ($2NH$) y real, por lo que la SVD es menos costosa y real.

$$\mathtt{A}_{\mathbb{R}} = \mathrm{Re}\,(\text{filas impares de } \mathtt{A}) = \mathrm{Re}\,(\text{filas pares de } \mathtt{A})$$

Cada terna de puntos (cada horóptera) contribuye con dos filas a la matriz $\mathtt{A}_{\mathbb{R}}$. Para resolver el sistema homogéneo hacen falta, al menos 5 filas, por eso son necesarias 3 horópteras como mínimo para estimar la cónica.

7.8.5.4. Equivalencia de los enfoques

Siguiendo un argumento similar al expuesto en § 6.2.3.2 y razonando sobre ambas matrices reales, es decir, utilizando las simplificaciones $\mathtt{A} = \mathtt{A}_{\mathbb{R}}$ y (7.29), se verifica:

$$\mathtt{M} = \mathtt{A}^{\top}\mathtt{A} \tag{7.30}$$

Desarrollemos la expresión (7.27) de una forma similar a la demostración en § 7.8.5.2, sin perder de vista la restricción $\|\mathbf{c}\| = 1$.

$$\begin{aligned}
&\mathbf{c}^{\top}\mathtt{M}\mathbf{c} && = \\
&\sum_{k=1}^{NH}\sum_{i=1}^{2}\left(|\mathbf{q}_{ik}^{\top}C\mathbf{q}_{ik}|^2 + |\mathbf{p}_k^{\top}C\mathbf{q}_{ik}|^2\right) && = \\
&\sum_{k=1}^{NH}\sum_{i=1}^{2}\left(|\mu_{\mathbf{q}_{ik}\,\mathbf{q}_{ik}}^{\top}\mathbf{c}|^2 + |\mu_{\mathbf{p}_k\,\mathbf{q}_{ik}}^{\top}\mathbf{c}|^2\right) && = \\
&\sum_{k=1}^{NH}\|\mathtt{A}_k\mathbf{c}\|^2 && = \\
&\|\mathtt{A}\mathbf{c}\|^2 && = \\
&\mathbf{c}^{\top}\mathtt{A}^{\top}\mathtt{A}\mathbf{c}
\end{aligned} \tag{7.31}$$

7.8.6. Algoritmo 3: horópteras y la Cuádrica Absoluta Dual

En esta sección se explica el algoritmo dual al de la sección § 7.8.5. Aunque no está descrito en ningún artículo, está inspirado en el trabajo de Ronda, Valdés y Jaureguizar sobre las horópteras [9].

Se asume, al igual que antes, que se parte de una calibración proyectiva de las cámaras, todas con los mismos parámetros intrínsecos $\mathtt{K}$ y distintos parámetros extrínsecos.

En lugar de estimar la cónica absoluta Ω_∞, se estima su dual: la cuádrica absoluta dual, $\mathtt{Q}_\infty^*$. Los fundamentos del algoritmo son los mismos: las propiedades de los puntos de corte de las horópteras entre cada par de cámaras con el plano del infinito, pero aplicados a la cuádrica absoluta dual. Las rectas que unen un punto de corte real y uno complejo, $V(\mathbf{p}, \mathbf{q}_1)$ y $V(\mathbf{p}, \mathbf{q}_2)$ son las bases de sendos haces de planos pertenecientes a la cuádrica absoluta dual. Además, el plano del infinito es el núcleo de la cuádrica absoluta dual.

$$\mathtt{Q}_\infty^*\pi_\infty = 0 \tag{7.32}$$

Estimaremos la cuádrica que mejor se ajusta según las anteriores condiciones. Además, se debe verificar otra restricción: la matriz de $\mathtt{Q}_\infty^*$ debe ser semidefinida positiva (o negativa, según el factor de escala): un autovalor nulo y el resto, positivos.

El problema es: dado un plano π^0 y varias ternas de puntos en él (uno real y dos complejos conjugados), hallar los haces de planos con base las rectas $V(\mathbf{p}, \mathbf{q}_{ik}), i = 1\ldots 2, k = 1\ldots NH$. Calcular la cuádrica $\mathtt{Q}^*$ que mejor se ajusta a dichos haces de planos sujeta a la restricción $\mathtt{Q}^*\pi^0 = 0$.

El ajuste de la cuádrica es un algoritmo lineal (DLT o SVD) con restricciones, en concreto y como se verá más adelante, el enfoque SVD encaja con el algoritmo A3.6, [1, pág. 565].

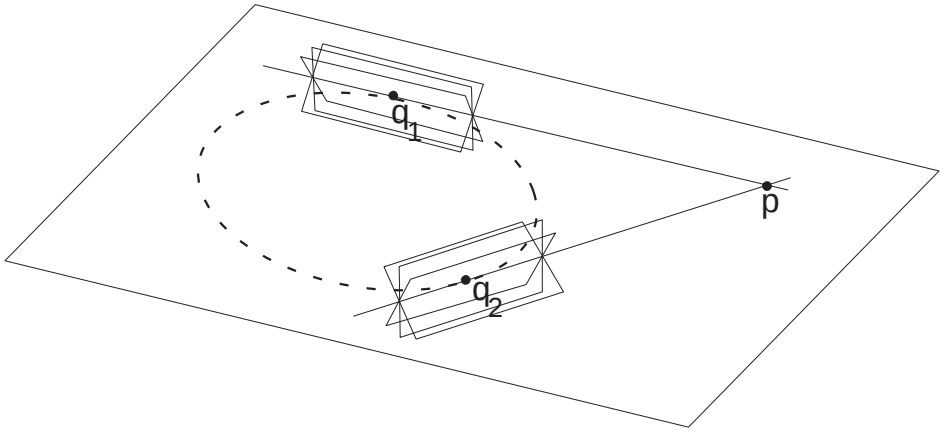


Figura 7.7: Configuración básica de una terna de puntos de corte y haces de planos, respecto de la cuádrica a estimar

7.8.6.1. Explicación del ajuste

En $\mathbb{P}^3$, una cuádrica de planos está representada por una matriz $\mathtt{Q}^*$ simétrica y de 4×4.

$$\mathtt{Q}^* = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ q_{14} & q_{24} & q_{34} & q_{44} \end{bmatrix}$$

La ecuación que verifica un plano π_1 que pertenece a la cuádrica es $\pi_1^\top \mathtt{Q}^* \pi_1 = 0$. Las coordenadas del punto polar de un plano π_1 son $\mathbf{X} = \mathtt{Q}^* \pi_1$, es decir, $\mathbf{X}^\top = \pi_1^\top \mathtt{Q}^{*\top} = \pi_1^\top \mathtt{Q}^*$ y la condición que verifica un plano π_1 que pasa por dicho punto $\mathbf{X}$ es $\mathbf{X}^\top \pi_1 = \pi_1^\top \mathtt{Q}^* \pi_1 = 0$.

El primer paso es obtener los haces de planos a partir de los puntos de corte de las horopteras con el plano. No hace falta realizar el paso intermedio de obtener las coordenadas de las rectas que son las bases de los haces, pues se pueden calcular directamente dos planos de cada haz.

Supongamos que tenemos una terna de puntos de corte, $(\mathbf{p}, \mathbf{q}_1, \mathbf{q}_2)$. La matriz W_1 de 2×4 con las coordenadas homogéneas de los puntos $\mathbf{p}$ y $\mathbf{q}_1$ tiene por núcleo dos planos del haz (π_{11}, π_{12}), cuya base es la recta $V(\mathbf{p}, \mathbf{q}_1)$. Estos planos se calculan mediante la SVD de W_1 (las dos últimas columnas de $\mathtt{V}$ en $W_1 = \mathtt{UDV}^\top$).

$$W_1 = \begin{bmatrix} \mathbf{p}^\top \\ \mathbf{q}_1^\top \end{bmatrix}$$

Es fácil de verificar que los planos pertenecen al haz, ya que $W_1 \pi_{11} = \mathbf{0}$ equivale expresar que el plano π_{11} pasa por los puntos $\mathbf{p}$, y $\mathbf{q}_1$, es decir, $\mathbf{p}^\top \pi_{11} = 0$ y $\mathbf{q}_1^\top \pi_{11} = 0$. El plano π_{12} también pasa por ambos puntos, ya que $W_1 \pi_{12} = \mathbf{0}$.

El mismo razonamiento es aplicable al otro haz de planos: el que se genera a partir de los puntos $\mathbf{p}$ y $\mathbf{q}_2$. Obtenemos dos planos (π_{21}, π_{22}) del haz a partir del núcleo de la matriz

$$W_2 = \begin{bmatrix} \mathbf{p}^\top \\ \mathbf{q}_2^\top \end{bmatrix}$$

Los dos haces de planos se pueden expresar como combinación lineal de los planos obtenidos:

$$\begin{aligned} \pi_1 &= \alpha_1 \pi_{11} + \beta_1 \pi_{12} \\ \pi_2 &= \alpha_2 \pi_{21} + \beta_2 \pi_{22} \end{aligned} \tag{7.33}$$

Todo plano de cada haz pertenece a la cuádrica que queremos estimar, por lo que se debe verificar:

$$\begin{array}{rclcl} \pi_1^\top \mathtt{Q}^* \pi_1 & = & 0 & \forall(\alpha_1,\beta_1) & \neq & (0,0) \\ \pi_2^\top \mathtt{Q}^* \pi_2 & = & 0 & \forall(\alpha_2,\beta_2) & \neq & (0,0) \end{array} \tag{7.34}$$

Como $\mathbf{q}_2 = \mathbf{q}_1^*$, se verifica

$$\begin{array}{rcl} \pi_{21} & = & \pi_{11}^* \\ \pi_{22} & = & \pi_{12}^* \end{array} \tag{7.35}$$

Un poco de notación:

La relación $\mathbf{X}^\top \mathtt{Q}^* \mathbf{Y} = 0$ se puede expresar como producto matricial de vectores.

Sean $\mathbf{X} = (X_1, X_2, X_3, X_4)^\top$, $\mathbf{Y} = (Y_1, Y_2, Y_3, Y_4)^\top$ coordenadas de puntos o planos en $\mathbb{P}^3$, y llamemos

$$\mu_{\mathbf{XY}} \equiv \mu(\mathbf{X}, \mathbf{Y}) = \begin{bmatrix} X_1Y_1 \\ X_1Y_2 + X_2Y_1 \\ X_1Y_3 + X_3Y_1 \\ X_1Y_4 + X_4Y_1 \\ X_2Y_2 \\ X_2Y_3 + X_3Y_2 \\ X_2Y_4 + X_4Y_2 \\ X_3X_3 \\ X_3Y_4 + X_4Y_3 \\ X_4Y_4 \end{bmatrix} \tag{7.36}$$

Además, se verifica

$$\mu(\mathbf{X}^*, \mathbf{Y}^*) = \mu^*(\mathbf{X}, \mathbf{Y}) \equiv \mu^*_{\mathbf{XY}} \tag{7.37}$$

Entonces, las siguientes condiciones son equivalentes:

$$\pi_i^\top \mathtt{Q}^* \pi_j = \mu_{\pi_i\,\pi_j}^\top \mathbf{q}^* = 0 \quad \forall(i,j)$$

$\mathbf{q}^* = [q_{11} \;\; q_{12} \;\; q_{13} \;\; q_{14} \;\; q_{22} \;\; q_{23} \;\; q_{24} \;\; q_{33} \;\; q_{34} \;\; q_{44}]^\top$ es la cuádrica, en forma de vector de 10×1.

7.8.6.2. Enfoque basado en autovalores

Con datos ruidosos, las condiciones (7.34) no se verificarán de forma exacta, por lo que en un primer momento se pensó en encontrar la cuádrica del siguiente problema de minimización:

$$\min_{\mathtt{Q}^*} \left(\|\mathtt{Q}^* \pi^0\|^2 + \sum_{k=1}^{NH} \sum_{i=1}^{2} |\pi_i^{k\top} \mathtt{Q}^* \pi_i^k|^2 \right) \Leftrightarrow \min_{\mathbf{q}^*} \left(\|\mathtt{A}_0 \mathbf{q}^*\|^2 + \sum_{k=1}^{NH} \sum_{i=1}^{2} |\mu_{\pi_i^k\,\pi_i^k}^\top \mathbf{q}^*|^2 \right)$$

Sin embargo, es fácil imponer la restricción de que el núcleo de la cuádrica estimada sea el plano de prueba ($\mathtt{Q}^* \pi^0$), por lo que la función de coste a minimizar cambia:

$$\min_{\mathtt{Q}^*} \sum_{k=1}^{NH} \sum_{i=1}^{2} |\pi_i^{k\top} \mathtt{Q}^* \pi_i^k|^2 \quad \text{sujeto a} \quad \mathtt{Q}^* \pi^0 = 0 \Leftrightarrow \min_{\mathbf{q}^*} \sum_{k=1}^{NH} \sum_{i=1}^{2} |\mu_{\pi_i^k\,\pi_i^k}^\top \mathbf{q}^*|^2 \quad \text{sujeto a} \quad \mathtt{A}_0 \mathbf{q}^* = 0 \tag{7.38}$$

Al igual que antes, NH es el número de horópteras. La matriz $\mathtt{A}_0$ se construye a partir de las coordenadas homogéneas del plano $\pi^0 = (\pi_1^0, \pi_2^0, \pi_3^0, \pi_4^0)^\top$.

$$\mathtt{A}_0 = \begin{bmatrix} \pi_1^0 & \pi_2^0 & \pi_3^0 & \pi_4^0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \pi_1^0 & 0 & 0 & \pi_2^0 & \pi_3^0 & \pi_4^0 & 0 & 0 & 0 \\ 0 & 0 & \pi_1^0 & 0 & 0 & \pi_2^0 & 0 & \pi_3^0 & \pi_4^0 & 0 \\ 0 & 0 & 0 & \pi_1^0 & 0 & 0 & \pi_2^0 & 0 & \pi_3^0 & \pi_4^0 \end{bmatrix} \tag{7.39}$$

Desarrollando la condición (7.38) para un haz de planos, llegamos a que se obtienen 2 condiciones de pertenencia y 2 de polaridad (en realidad una sola, que aparece dos veces) respecto de la cuádrica.

$$|\pi_1^\top \mathtt{Q}^* \pi_1|^2 = |(\alpha_1\pi_{11} + \beta_1\pi_{12})^\top \mathtt{Q}^* (\alpha_1\pi_{11} + \beta_1\pi_{12})|^2 = 0 \quad \forall (\alpha_1, \beta_1) \neq (0,0)$$

equivale proyectivamente a

$$|\pi_{11}^\top \mathtt{Q}^* \pi_{11}|^2 \quad = 0 \quad \text{si } \beta_1 = 0$$

$$|\pi_{12}^\top \mathtt{Q}^* \pi_{12}|^2 \quad = 0 \quad \text{si } \alpha_1 = 0$$

Las anteriores ecuaciones expresan que los planos que parametrizan el haz pertenecen a la cuádrica. Teniendo en cuenta estas expresiones, existen además relaciones de polaridad si $(\alpha_1, \beta_1) \sim (1, \rho_1)$:

$$|(\pi_{11} + \rho_1\pi_{12})^\top \mathtt{Q}^* (\pi_{11} + \rho_1\pi_{12})|^2 = |\rho_1|^2 \, |\pi_{11}^\top \mathtt{Q}^* \pi_{12} + \pi_{12}^\top \mathtt{Q}^* \pi_{11}|^2 = 4|\rho_1|^2 \, |\pi_{11}^\top \mathtt{Q}^* \pi_{12}|^2 = 0$$

Como se debe verificar $\forall \rho_1$, se deduce que se debe cumplir

$$|\pi_{11}^\top \mathtt{Q}^* \pi_{12}|^2 = 0 \tag{7.40}$$

En ausencia de ruido, las ecuaciones se verifican de manera exacta y no importa que la condición de polaridad $\pi_{11}^\top \mathtt{Q}^* \pi_{12}$ aparezca dos veces dentro de $\pi_1^\top \mathtt{Q}^* \pi_1$. Sin embargo, en presencia de ruido se ha decidido tener en cuenta esa doble aparición y así minimizar algo parecido a (7.38). Tener en cuenta este factor de una forma u otra es equivalente a modificar el problema para que sea de mínimos cuadrados ponderados.

$$\min_{\mathtt{Q}^*} |\pi_1^\top \mathtt{Q}^* \pi_1|^2 \quad \forall (\alpha_1, \beta_1) \neq (0,0) \rightsquigarrow \min_{\mathtt{Q}^*} \left(|\pi_{11}^\top \mathtt{Q}^* \pi_{11}|^2 + |\pi_{12}^\top \mathtt{Q}^* \pi_{12}|^2 + 2|\, \pi_{11}^\top \mathtt{Q}^* \pi_{12}|^2 \right)$$

Y se obtiene algo parecido a la expresión (7.38), pero separable

$$\min_{\mathbf{q}^*} \sum_{k=1}^{NH} \sum_{i=1}^{2} \left(\sum_{j=1}^{2} |\mu_{\pi_{ij}^k \, \pi_{ij}^k}^\top \mathbf{q}^*|^2 + 2|\mu_{\pi_{i1}^k \, \pi_{i2}^k}^\top \mathbf{q}^*|^2 \right) \quad \text{sujeto a} \quad \mathtt{A}_0 \mathbf{q}^* = 0 \tag{7.41}$$

Al igual que en el algoritmo dual, se puede desarrollar la expresión anterior para concluir que pretendemos minimizar un polinomio cuadrático $g(\mathbf{q}^*)$, pero esta vez sujeto a dos restricciones: $\mathtt{A}_0 \mathbf{q}^* = 0$.

$$\min_{\|\mathbf{q}^*\|=1} g(\mathbf{q}^*) \quad \text{sujeto a} \quad \mathtt{A}_0 \mathbf{q}^* = 0 \Leftrightarrow \min_{\|\mathbf{q}^*\|=1} \mathbf{q}^{*\top} \mathtt{M} \mathbf{q}^* \quad \text{sujeto a} \quad \mathtt{A}_0 \mathbf{q}^* = 0$$

Esta vez, $\mathtt{M}$ es una matriz simétrica de 10×10. Mediante el cambio de variable $\mathbf{q}^* = \mathtt{A}_0^\perp \mathbf{q}'^*$ se modifica el problema para que sea uno del estilo del resuelto en § 7.8.5.2. Los detalles de la solución que viene a continuación están justificados en la siguiente sección. Al realizar el cambio, la expresión anterior queda:

$$\min_{\|\mathbf{q}'^*\|=1} (\mathtt{A}_0^\perp \mathbf{q}'^*)^\top \mathtt{M} (\mathtt{A}_0^\perp \mathbf{q}'^*) = \min_{\|\mathbf{q}'^*\|=1} \mathbf{q}'^{*\top} (\mathtt{A}_0^\perp)^\top \mathtt{M} \mathtt{A}_0^\perp \mathbf{q}'^* = \min_{\|\mathbf{q}'^*\|=1} \mathbf{q}'^{*\top} \mathtt{M}' \mathbf{q}'^*$$

siendo $\mathtt{M}' = (\mathtt{A}_0^\perp)^\top \mathtt{M} \mathtt{A}_0^\perp$. En lugar de $\mathtt{M}$ y la restricción de $\mathtt{A}_0$ se obtiene una matriz $\mathtt{M}'$, que incluye la restricción.

Este problema ya lo sabemos resolver, pues es como el de § 7.8.5.2. La función de Lagrange es:

$$F(\mathbf{q}'^*, z) = \mathbf{q}'^{*\top} \mathtt{M}' \mathbf{q}'^* + z(\mathbf{q}'^{*\top} \mathbf{q}'^* - 1)$$

Si se iguala a cero su gradiente, se obtiene un sistema de ecuaciones que indica los puntos críticos.

$$\mathbf{0} = \nabla F = \begin{cases} \dfrac{\partial F}{\partial \mathbf{q}'^*} & = & \dfrac{\partial}{\partial \mathbf{q}'^*} \left(\mathbf{q}'^{*\top} \mathtt{M}' \mathbf{q}'^* \right) + z \dfrac{\partial}{\partial \mathbf{q}'^*} \left(\mathbf{q}'^{*\top} \mathbf{q}'^* - 1 \right) = 2\mathtt{M}' \mathbf{q}'^* + 2z\mathbf{q}'^* \\ \dfrac{\partial F}{\partial z} & = & \mathbf{q}'^{*\top} \mathbf{q}'^* - 1 \end{cases}$$

De la primera ecuación resulta el sistema

$$\mathtt{M}'\mathbf{q}'^{*} = -z\mathbf{q}'^{*} \tag{7.42}$$

de donde se deduce que $\mathbf{q}'^{*}$ es un autovector de $\mathtt{M}'$ con autovalor $\lambda = -z$. Se escoge el autovalor que minimiza el coste, es decir, el menor autovalor:

$$\mathbf{q}'^{*\top}\mathtt{M}'\mathbf{q}'^{*} = \mathbf{q}'^{*\top}(-z\mathbf{q}'^{*}) = \lambda\mathbf{q}'^{*\top}\mathbf{q}'^{*} = \lambda$$

Solución: $\mathbf{q}'^{*}$ es el autovector de $\mathtt{M}'$ correspondiente al menor autovalor. El coste del ajuste es el menor autovalor de $\mathtt{M}'$. Una vez obtenido $\mathbf{q}'^{*}$, se calcula $\mathbf{q}^{*}$ a partir del cambio de variable y queda estimada la cuádrica.

DEMO

Desarrollemos la segunda parte de la expresión (7.41) y tengamos en cuenta que buscamos una cuádrica de coeficientes reales, por lo que el conjugado del vector $\mathbf{q}^{*}$ es él mismo. No confundir el asterisco del vector $\mathbf{q}^{*}$, que significa "dual" con el asterisco de los vectores μ, que significan "conjugado".

$$\begin{aligned}
&\sum_{k=1}^{NH}\sum_{i=1}^{2}\left(\sum_{j=1}^{2}|\mu^{\top}_{\pi^k_{ij}\,\pi^k_{ij}}\mathbf{q}^{*}|^2 + 2|\mu^{\top}_{\pi^k_{i1}\,\pi^k_{i2}}\mathbf{q}^{*}|^2\right) &= \\
&\sum_{k=1}^{NH}\sum_{i=1}^{2}\left(\sum_{j=1}^{2}(\mu^{\top}_{\pi^k_{ij}\,\pi^k_{ij}}\mathbf{q}^{*})(\mu^{*\top}_{\pi^k_{ij}\,\pi^k_{ij}}\mathbf{q}^{*}) + 2(\mu^{\top}_{\pi^k_{i1}\,\pi^k_{i2}}\mathbf{q}^{*})(\mu^{*\top}_{\pi^k_{i1}\,\pi^k_{i2}}\mathbf{q}^{*})\right) &= \\
&\sum_{k=1}^{NH}\sum_{i=1}^{2}\left(\sum_{j=1}^{2}(\mathbf{q}^{*\top}\mu_{\pi^k_{ij}\,\pi^k_{ij}})(\mu^{*}_{\pi^k_{ij}\,\pi^k_{ij}}\mathbf{q}^{*}) + 2(\mathbf{q}^{*\top}\mu_{\pi^k_{i1}\,\pi^k_{i2}})(\mu^{*\top}_{\pi^k_{i1}\,\pi^k_{i2}}\mathbf{q}^{*})\right) &= \\
&\sum_{k=1}^{NH}\sum_{i=1}^{2}\left(\sum_{j=1}^{2}(\mathbf{q}^{*\top}\mathtt{M}_{\pi^k_{ij}\,\pi^k_{ij}}\mathbf{q}^{*}) + 2(\mathbf{q}^{*\top}\mathtt{M}_{\pi^k_{i1}\,\pi^k_{i2}}\mathbf{q}^{*})\right) &= \\
&\mathbf{q}^{*\top}\left(\sum_{k=1}^{NH}\sum_{i=1}^{2}\left(\sum_{j=1}^{2}\mathtt{M}_{\pi^k_{ij}\,\pi^k_{ij}} + 2\mathtt{M}_{\pi^k_{i1}\,\pi^k_{i2}}\right)\right)\mathbf{q}^{*} &= \\
&\mathbf{q}^{*\top}\left(\sum_{k=1}^{NH}\mathtt{M}_k\right)\mathbf{q}^{*} &= \\
&\mathbf{q}^{*\top}\mathtt{M}\mathbf{q}^{*}
\end{aligned}$$

Hemos llamado

$$\begin{aligned}
\mu_{\pi^k_{ij}\,\pi^k_{ij}}\,\mu^{*\top}_{\pi^k_{ij}\,\pi^k_{ij}} &= \mathtt{M}_{\pi^k_{ij}\,\pi^k_{ij}} \\
\mu_{\pi^k_{i1}\,\pi^k_{i2}}\,\mu^{*\top}_{\pi^k_{i1}\,\pi^k_{i2}} &= \mathtt{M}_{\pi^k_{i1}\,\pi^k_{i2}}
\end{aligned}$$

Además, se verifica que,

$$\begin{aligned}
\mu_{\pi^k_{2j}\,\pi^k_{2j}} &= \mu^{*}_{\pi^k_{1j}\,\pi^k_{1j}} \\
\mu_{\pi^k_{21}\,\pi^k_{22}} &= \mu^{*}_{\pi^k_{11}\,\pi^k_{12}}
\end{aligned}$$

que implica

$$\begin{aligned}
\mathtt{M}_{\pi^k_{2j}\,\pi^k_{2j}} &= \mathtt{M}^{*}_{\pi^k_{1j}\,\pi^k_{1j}} \\
\mathtt{M}_{\pi^k_{21}\,\pi^k_{22}} &= \mathtt{M}^{*}_{\pi^k_{11}\,\pi^k_{12}}
\end{aligned}$$

por lo que

$$\mathtt{M}_k = \sum_{i=1}^{2}\left(\sum_{j=1}^{2}\mathtt{M}_{\pi_{ij}^k\,\pi_{ij}^k} + 2\mathtt{M}_{\pi_{i1}^k\,\pi_{i2}^k}\right) = 2\,\mathrm{Re}\left[\mathtt{M}_{\pi_{11}^k\,\pi_{11}^k} + \mathtt{M}_{\pi_{12}^k\,\pi_{12}^k} + 2\mathtt{M}_{\pi_{11}^k\,\pi_{12}^k}\right]$$

y la matriz $\mathtt{M}$ queda, al final:

$$\mathtt{M} = \sum_{k=1}^{NH}\mathtt{M}_k = \sum_{k=1}^{NH} 2\,\mathrm{Re}\left[\mathtt{M}_{\pi_{11}^k\,\pi_{11}^k} + \mathtt{M}_{\pi_{12}^k\,\pi_{12}^k} + 2\mathtt{M}_{\pi_{11}^k\,\pi_{12}^k}\right] \tag{7.43}$$

Aunque es muy difícil de ver, Se puede interpretar geométricamente que $\mathtt{M}$ es la matriz de una cuádrica en $\mathbb{P}^9$. Y seguir con el mismo razonamiento que el dado en § 7.8.5.2.

7.8.6.3. Enfoque SVD

Otro enfoque equivalente consiste en poner las condiciones que deben verificarse una a continuación de otra en forma matricial (condiciones lineales). Si lo hacemos para todas las ternas de puntos, obtendremos la matriz de diseño $\mathtt{A}$, de tamaño $6NH \times 10$, del sistema homogéneo $\mathtt{A}\mathbf{q}^* = \mathbf{0}$.

$$\mathtt{A}_k = \begin{bmatrix} \mu_{\pi_{11}^k\,\pi_{11}^k}^\top \\ \mu_{\pi_{12}^k\,\pi_{12}^k}^\top \\ \sqrt{2}\mu_{\pi_{11}^k\,\pi_{12}^k}^\top \\ \mu_{\pi_{21}^k\,\pi_{21}^k}^\top \\ \mu_{\pi_{22}^k\,\pi_{22}^k}^\top \\ \sqrt{2}\mu_{\pi_{21}^k\,\pi_{22}^k}^\top \end{bmatrix} \qquad \mathtt{A} = \begin{bmatrix} \mathtt{A}_1 \\ \mathtt{A}_2 \\ \vdots \\ \mathtt{A}_{NH} \end{bmatrix}$$

En el caso de datos exactos ruidosos, no se verificará la anterior ecuación matricial, sino que habrá un vector de error o residuo $\mathtt{A}\mathbf{q}^* = \epsilon$, y sería deseable minimizar la norma de dicho vector de error, a la vez que imponer la condición del núcleo de la cuádrica. Aunque es también una condición lineal, no se añade a las filas de la matriz $\mathtt{A}$ para minimizar su módulo, sino que queremos que siempre se verifique. El problema con restricciones resultante es:

$$\min_{\|\mathbf{q}^*\|=1} \|\mathtt{A}\mathbf{q}^*\|^2 \quad \text{sujeto a} \quad \mathtt{A}_0\mathbf{q}^* = 0$$

La solución de este problema encaja con el algoritmo A3.6, [1, pág. 565]. En este caso, la matriz con restricciones es $\mathtt{A}_0$, 4×10 y de rango 4. Esta matriz tiene un núcleo de dimensión 6, así que para que se verifique la restricción se expresa la solución $\mathbf{q}^*$ como combinación lineal de una base de ese núcleo: $\mathbf{q}^* = \mathtt{A}_0^\perp \mathbf{q}'^*$. La matriz $\mathtt{A}_0^\perp$ es 10×6 y sus columnas son 6 vectores ortonormales que forman la base del núcleo de $\mathtt{A}_0$, generan el llamado complemento ortogonal de $\mathtt{A}_0$ y se obtienen mediante la SVD de $\mathtt{A}_0$. Es de notar que $\|\mathbf{q}^*\| = \|\mathbf{q}'^*\|$ porque $\mathtt{A}_0^\perp$ es una matriz con columnas ortogonales $(\mathtt{A}_0^\perp)^\top \mathtt{A}_0^\perp = \mathtt{I}_{6\times 6}$, con lo que el problema tiene todos los ingredientes del clásico:

$$\min_{\|\mathbf{q}'^*\|=1} \|\mathtt{A}\mathtt{A}_0^\perp \mathbf{q}'^*\|^2$$

La solución de éste último se obtiene mediante la SVD de $\mathtt{A}\mathtt{A}_0^\perp = \mathtt{U}\mathtt{D}\mathtt{V}^\top$. El vector $\mathbf{q}'^*$ es la columna de $\mathtt{V}$ correspondiente al menor valor singular de $\mathtt{A}\mathtt{A}_0^\perp$. Una vez obtenido $\mathbf{q}'^*$, el vector $\mathbf{q}^*$ se obtiene a través de la matriz $\mathtt{A}_0^\perp$.

Como se ha mostrado, cada matriz $\mathtt{A}_k$ es compleja, con las tres últimas filas conjugadas de las tres primeras. Se puede simplificar más la matriz $\mathtt{A}$ explotando esta propiedad de conjugación, para obtener

una matriz equivalente $\mathtt{A}_{\mathbb{R}}$, pero de la mitad de filas ($3NH$) y real, por lo que la SVD es menos costosa y real. Simplemente habría que sustituir $\mathtt{A}_k$ en las expresiones anteriores por:

$$\mathtt{A}_k = \mathrm{Re}\begin{bmatrix} \mu_{\pi_{11}^k}^{\top}\pi_{11}^k \\ \mu_{\pi_{12}^k}^{\top}\pi_{12}^k \\ \sqrt{2}\mu_{\pi_{11}^k}^{\top}\pi_{12}^k \end{bmatrix}$$

7.8.6.4. Equivalencia de los enfoques

De manera similar a lo explicado en § 7.8.5.4, ambos enfoques para la cuádrica absoluta dual son equivalentes. Si se consideran las matrices reales $\mathtt{M}$ y $\mathtt{A}$ ($\mathtt{A}_k$ de 3 filas), entonces se verifica la misma relación que para el algoritmo de estimación de la cónica: (7.30).

La equivalencia de los enfoques se ha utilizado para dar la solución del enfoque de autovalores una vez conocida la solución del enfoque SVD.

7.8.6.5. Otros comentarios

Una ventaja de este algoritmo es que estima la cuádrica absoluta dual, un objeto propio del espacio proyectivo $\mathbb{P}^3$, utilizando todas las coordenadas de los puntos de corte de las horópteras con el plano del infinito. De los haces de planos obtenidos de estos puntos también se utilizan todas las coordenadas. No es necesario poner coordenadas homogéneas en una variedad de $\mathbb{P}^3$ o proyectar los puntos al plano de la retina, como se hace al estimar la cónica absoluta o su imagen (IAC), respectivamente. El objeto que queremos estimar, $\mathtt{Q}_{\infty}^{*}$, no está contenido en ninguna variedad lineal del ambiente, al contrario que la cónica absoluta Ω_{∞} o la IAC ω, que cada una está contenida en un hiperplano de $\mathbb{P}^3$. A pesar de todo, según se demostrará en las pruebas experimentales, es mejor trabajar en las imágenes, como realiza el algoritmo 2, con las IACs.

Se ha comprobado que no es fácil imponer penalizaciones en las funciones de coste y que el algoritmo LM converja tan rápidamente como si no estuvieran. Por ejemplo, en el caso del algoritmo 2 de estimación de la PAC, no se penaliza la solución que no es una matriz definida; ni en el algoritmo 3 se penaliza una cuádrica solución que no sea de rango tres o no tenga todos sus autovalores del mismo signo.

7.9. Calibration Pencil

En esta sección se estudian distintas posibilidades de recuperar la estructura euclídea del espacio mediante el *Calibration Pencil*, cuya definición y caracterización se encuentran en [10]. También se hace referencia al artículo precedente relacionado, [11]. La figura 4.3 es un dibujo ilustrativo de las rectas que se utilizan para definir el Calibration Pencil.

Prácticamente todo el contenido de la sección son estudios sobre algoritmos no lineales de estimación del Calibration Pencil, sin previamente presentar sus características. Se debe mencionar que esta sección no es tan fructífera en resultados prácticos como la de las horópteras ya que ninguno de los algoritmos no lineales propuestos da mejores resultados que el algoritmo lineal de [10], sin embargo el esfuerzo mental invertido en tratar de mejorar la estimación lineal no es despreciable.

Definición y restricciones

El Calibration Pencil es un haz de cuádricas en $\mathbb{P}^5$ que contiene toda la información para realizar una calibración euclídea a partir de una calibración proyectiva de al menos 10 imágenes, en el caso de cámaras con píxeles de forma conocida. Como ya se ha mencionado, en [10] se expone un algoritmo

lineal para estimar dicho objeto, basado en la solución por mínimos cuadrados sin imponer las restricciones sobre ciertas propiedades características que debería cumplir dicho haz de cuádricas.

El haz de cuádricas lo podemos representar mediante la ecuación matricial

$$\Sigma' = a\Sigma + b\Omega \tag{7.44}$$

siendo Ω la cuádrica de Klein y Σ una cuádrica de rango 3. Ambas cuádricas ya fueron introducidas en § 3.4.5 y la ecuación (7.44) es la expresión del haz de cuádricas en función de dos de ellas. La cuádrica de Klein tiene una matriz conocida, de rango máximo e invariante ante transformaciones en $\mathbb{P}^5$ provenientes del grupo de las transformaciones proyectivas de $\mathbb{P}^3$, ergo no hace falta estimarla. Sin embargo, la cuádrica degenerada Σ no es invariante ante dichas transformaciones, por lo que sí es necesario estimarla.

Las restricciones principales que debe verificar la cuádrica Σ es que su rango sea menor o igual que 3 y que todos sus autovalores no nulos sean del mismo signo (semidefinida positiva o negativa). Además, la cuádrica de Klein y Σ son "ortogonales".

Existen diversas formas de expresar la condición de rango 3, como se indica en los dos artículos antes citados. Sin embargo, el algoritmo lineal de [10] no impone dicha restricción, sino que los tres menores valores singulares son anulados a posteriori. Es bien conocido que esta acción se puede interpretar como proyectar la cuádrica obtenida sobre el espacio de las cuádricas de rango 3, en el sentido de que la cuádrica resultado es la más cercana a la cuádrica inicial en la norma inducida, al igual que se hace en el algoritmo de los 8 puntos para la matriz fundamental. Esta forma de imponer la condición de rango 3 no tiene sencilla interpretación y no es cierto que la cuádrica obtenida sea la mejor en el sentido de mínimos cuadrados.

Merece la pena observar que dicho algoritmo lineal sí impone la condición de ortogonalidad con la cuádrica de Klein.

7.9.1. Algoritmos no lineales de estimación

El paso siguiente consiste en tratar de mejorar la estimación de la cuádrica Σ mediante algún procedimiento no lineal que imponga las restricciones propias del problema. Lo primero que viene a la mente al pensar en problemas de optimización con restricciones es la herramienta matemática de los multiplicadores de Lagrange. Leyendo otros artículos [15], tropezamos con una técnica llamada Sequential Quadratic Programming, en adelante SQP, la cual se emplea para el tipo de problemas que queremos resolver. Así, pues, en la siguiente sección explicaremos cómo formular nuestro problema de estimación del Calibration Pencil según el enfoque de la SQP.

7.9.1.1. Algoritmos basados en la SQP

Elementos de un problema que utiliza la SQP

La programación secuencial cuadrática (SQP) es un esquema general de optimización de funciones de coste no lineales y suaves sujetas a restricciones también no lineales y suaves; es del estilo del algoritmo de Newton en el sentido de que necesita las segundas derivadas de la función de coste y potencialmente ofrece convergencia cuadrática. La versión que daremos sólo impone restricciones de igualdad, existen versiones más elaboradas que permiten trabajar con restricciones de desigualdad, además de ofrecer estabilidad y esquemas de control del paso. En [15], se expone un breve resumen el algoritmo de optimización con restricciones utilizado.

El objetivo es minimizar una función de coste escalar $f(\mathbf{x})$ sujeta a la ecuación vector de restricciones $\mathbf{c}(\mathbf{x}) = \mathbf{0}$. Los multiplicadores de Lagrange $\mathbf{z}$ proporcionan una solución implícita. La función de

Lagrange es:

$$F(\mathbf{x},\mathbf{z}) = f(\mathbf{x}) + \mathbf{z}^\top \mathbf{c}(\mathbf{x})$$

Al igualar su gradiente a cero (respecto a $\mathbf{x}$ y a $\mathbf{z}$) se obtiene un sistema de ecuaciones que indica los puntos críticos candidatos a minimizar f sujeta a las restricciones $\mathbf{c}$.

$$\mathbf{0} = \nabla F = \begin{cases} \dfrac{\partial F}{\partial \mathbf{x}} &= \nabla f(\mathbf{x}) + \mathbf{z}^\top \nabla \mathbf{c}(\mathbf{x}) \\ \dfrac{\partial F}{\partial \mathbf{z}} &= \mathbf{c}(\mathbf{x}) \end{cases}$$

Es de notar que $\nabla \mathbf{c}$ es la matriz jacobiana de la función multidimensional definida por el vector de restricciones. Equivalente a lo anterior, pero en una línea:

$$\nabla f + \mathbf{z}^\top \nabla \mathbf{c} = \mathbf{0} \qquad \text{con} \qquad \mathbf{c}(\mathbf{x}) = \mathbf{0}$$

El método consiste en resolver esto iterativamente, comenzado con una estimación inicial $\mathbf{x}_0$. Se utiliza la aproximación cuadrática de Taylor de la función de coste y la aproximación lineal de la restricción, ambas evaluadas en $\mathbf{x}_0$, lo que lleva a un subproblema de optimización cuadrático con restricciones lineales:

$$\min_{\delta \mathbf{x}} \left(\delta \mathbf{x}^\top \nabla f + \frac{1}{2} \delta \mathbf{x}^\top \nabla^2 f \, \delta \mathbf{x} \right) \Big|_{\mathbf{c} + \delta \mathbf{x}^\top \nabla \mathbf{c} = \mathbf{0}}$$

siendo $\nabla^2 f = \mathtt{H}$ el Hessiano de f, evaluado en $\mathbf{x}_0$.

Este subproblema tiene una solución lineal exacta:

$$\begin{pmatrix} \nabla^2 f & \nabla \mathbf{c} \\ \nabla \mathbf{c}^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} \delta \mathbf{x} \\ \mathbf{z} \end{pmatrix} = - \begin{pmatrix} \nabla f \\ \mathbf{c} \end{pmatrix}$$

Resolver para $\delta \mathbf{x}$, actualizar $\mathbf{x}_0$ a $\mathbf{x}_1 = \mathbf{x}_0 + \delta \mathbf{x}$, re-estimar las derivadas e iterar hasta que converja.

En nuestro caso, la función de coste a minimizar es el coste algebraico definido por la fórmula (7.45). No hace falta incluir los términos de coste sobre la cuádrica de Klein, ya que las coordenadas de Plücker de una recta en $\mathbb{P}^3$ siempre pertenecen a la cuádrica de Klein ($\hat{r}_i^\top \Omega \hat{r}_i = 0$).

$$\text{coste} = \sum_{i=1}^{N} |\hat{r}_i^\top \hat{\Sigma} \hat{r}_i|^2 = \sum_{i=1}^{N} |\frac{r_i^\top}{\|r_i\|} \frac{\Sigma}{\|\Sigma\|_F} \frac{r_i}{\|r_i\|}|^2 = \sum_{i=1}^{N} \frac{|r_i^\top \Sigma r_i|^2}{\|r_i\|^2 \|\Sigma\|_F} \tag{7.45}$$

siendo $\hat{r}_i$ el vector unitario con las coordenadas de Plücker de la recta i-ésima, y $\hat{\Sigma}$ la matriz de la cuádrica normalizada a norma Frobenius unidad. Así, el coste es independiente del escalado típico de las coordenadas homogéneas.

A partir de esta función de coste deben ser calculado el gradiente y el Hessiano. En la mayoría de las ocasiones se estimarán de forma numérica, no analíticamente, por rapidez en la implementación.

Los otros elementos necesarios son el vector de restricciones y su matriz jacobiana. La mayoría de las restricciones son polinómicas, por lo que las derivadas son sencillas de calcular analíticamente.

Para el Calibration Pencil, hay dos restricciones que son comunes a los tres siguientes desarrollos:

- La condición que elimina el factor de escala, por ejemplo $\|\Sigma\|^2 = 3$.

$$c(1) = \|\Sigma\|^2 - 3$$

- La condición de ortogonalidad con la cuádrica de Klein. Ésta es una restricción lineal,$\Sigma_{16} - \Sigma_{25} + \Sigma_{34} = 0$, de la cual se podría despejar alguno de sus elementos en función de los otros dos, y sustituir en el problema, para así asegurarse que siempre se verifica la condición. El signo menos es debido a que la cuádrica de Klein tiene los elementos 2 y 5 de su antidiagonal negativos:

$$\Omega = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$c(2) = \Sigma_{16} - \Sigma_{25} + \Sigma_{34}$$

Lo que difiere de una formulación a otra es la forma de expresar la condición de rango 3.

Restricción: SVD

Se impone la restricción de rango 3 de la forma más directa, utilizando la Descomposición en Valores Singulares, $\Sigma = \mathtt{UDU}^\top$, siendo $\mathtt{D} = \mathrm{diag}(\sigma_1 \dots \sigma_6)$. Los menores tres valores singulares deben ser nulos:

$$\begin{aligned} c(3) &= \sigma_4 \\ c(4) &= \sigma_5 \\ c(5) &= \sigma_6 \end{aligned}$$

La matriz jacobiana de estas restricciones se evalúa de forma numérica, debido a la ignorancia del autor para calcularlas analíticamente. Así que se espera que las restricciones 3 a 5 tengan derivadas continuas de segundo orden, para que encajen dentro de la teoría de la SQP.

Restricción: $\Sigma\Omega\Sigma = 0$

En [10] se demuestra que la condición de rango 3 es equivalente a la condición $\Sigma\Omega\Sigma = 0$, de forma que se puede expresar mediante 21 ecuaciones cuadráticas (grado 2) en los elementos de Σ. Aunque Ponce [11] dice que sólo hay 12 ecuaciones linealmente independientes, sin demostrarlo explícitamente. Se ha decidido incluir las 21 ecuaciones por si acaso, para evitar situaciones degeneradas. Las segundas derivadas de las ecuaciones de las restricciones son continuas, pues son nulas, lo que justifica que encajen dentro de la teoría de la SQP.

La matriz jacobiana de las 21 ecuaciones se calcula de forma analítica, lo que hace la ejecución es más rápida y exacta.

Restricción: menores de orden 4

Inicialmente se pensó en imponer la restricción de rango 3 a lo bruto, es decir, imponiendo que todos los determinantes de orden 4 de la matriz Σ fuesen nulos. ¿Cuántos menores hay? Los menores se obtienen de la matriz de 6×6 quitando dos filas y dos columnas, sin importar el orden. Por lo que hay $\binom{6}{4}^2 = 225$ menores. Explotando la simetría, el número de determinantes distintos se puede reducir aproximadamente a la mitad: $(225 - 15)/2 + 15 = 120$. Esto es así porque el determinante del menor formado por las filas y columnas $(1, 2, 3, 4) \times (1, 2, 3, 5)$ es el mismo que el menor formado con las filas y columnas intercambiadas: $(1, 2, 3, 5) \times (1, 2, 3, 4)$, por lo que se puede prescindir de uno de los dos. En cambio, no se puede prescindir de los menores con simetría, por ejemplo $(2, 3, 5, 6) \times (2, 3, 5, 6)$, de los cuales hay 15.

Lo poco atractivo de este desarrollo es que hay 120 ecuaciones de grado 4 (muchas derivadas y más complicadas), mientras que en el desarrollo anterior hemos visto que hay 21 ecuaciones de grado 2. Debido a esta comparación, se decidió no implementar las 120 ecuaciones.

7.9.1.2. Algoritmos basados en una parametrización

La matriz Σ admite una parametrización mediante una matriz R de 3×6, tal que $\Sigma = R^\top R$. Podemos pensar en optimizar respecto de los 18 elementos que forman R e imponiendo una restricción que elimine el factor de escala, por ejemplo $\|R\|_F = 1$. Tanto R como su opuesta $-R$ conducen a la misma Σ. Es más, cualquier matriz $R' = R\mathtt{U}$, siendo $\mathtt{U}$ una matriz ortogonal de 3×3 conduce a la misma Σ, por lo que existe una limitación, parecida a la de la calibración proyectiva.

El caso es que si variamos los elementos de R jamás podremos salirnos del espacio de las cuádricas de $\mathbb{P}^5$ de rango menor o igual que 3. No nos saldremos de esa variedad, ergo el vector de parámetros que minimice la función de coste, cumplirá la restricción $\text{rango}(\Sigma) \leq 3$.

7.9.2. La restricción τ y θ conocidos

El algoritmo de autocalibración basado en el Calibration Pencil se asienta sobre la suposición de conocer la relación de aspecto τ y el ángulo θ entre los lados de los píxeles (skew).

Es más, es posible restringir el análisis al caso de píxeles cuadrados ($\tau = 1$ y $\theta = \pi/2$), ya que si son conocidos τ y θ es aplicable una transformación afín a las imágenes para convertir los píxeles en cuadrados, antes de realizar la autocalibración. Dicha transformación afín es

$$\mathtt{H}(\tau,\theta) = \begin{bmatrix} 1 & \tau\cos\theta & 0 \\ 0 & \tau\,\text{sen}\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.46}$$

De tal forma que las coordenadas corregidas $\mathbf{x}_c$ cumplen la restricción de píxeles cuadrados:

$$\mathbf{x}_c = \mathtt{H}(\tau,\theta)\mathbf{x},$$

porque

$$\mathtt{K}_c = \mathtt{H}(\tau,\theta)\mathtt{K} = \begin{bmatrix} 1 & \tau\cos\theta & 0 \\ 0 & \tau\,\text{sen}\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_u & -\alpha_u\cot\theta & u_0 \\ 0 & \alpha_v/\,\text{sen}\,\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0' \\ 0 & \alpha_u & v_0' \\ 0 & 0 & 1 \end{bmatrix}$$

es una matriz de parámetros intrínsecos con píxeles cuadrados, donde $u_0' = u_0 - v_0\tau\cos\theta$ y $v_0' = v_0\tau\,\text{sen}\,\theta$ son las nuevas coordenadas del punto principal.

7.10. Evaluación experimental

Después de ver que existen numerosos algoritmos de autocalibración, el conjunto de pruebas posibles a realizar es enorme. Por ejemplo, un mismo algoritmo no lineal admite distintas inicializaciones, por lo que se puede evaluar su rendimiento en función de dicha inicialización. En la mayoría de las pruebas realizadas utilizaremos la suposición de cámaras ortogonales y su algoritmo correspondiente para inicializar el resto de algoritmos más complejos.

Presentaremos unas pruebas sencillas, pero no triviales. Están orientadas a evaluar los principales algoritmos de estimación de los parámetros intrínsecos de las cámaras en las condiciones más favorables, lo que nos proporciona una idea de su rendimiento.

La simulación consiste en la generación de cámaras y puntos 3D de la misma manera que se hizo para evaluar el rendimiento de la calibración proyectiva multicámara, que a su vez está basada en los experimentos de la matriz fundamental. El número de cámaras simulado es $m = 5$, todas con los mismos parámetros intrínsecos. En un primer conjunto de pruebas, las cámaras poseen una distancia focal de 20 mm y punto principal en el origen de coordenadas, además de píxeles cuadrados (suposición que

mantendremos en todos los experimentos). En un segundo conjunto de experimentos cambiaremos la distancia focal a 30 mm. Por último variaremos la distancia focal y el punto principal.

Los centros ópticos de las cámaras están situados sobre una superficie esférica de radio 8 m, concéntrica con una esfera de puntos a la que están apuntando. Los puntos están uniformemente distribuidos dentro de una esfera de radio 1 m.

El número de puntos (n) utilizado es variable, entre 20 y 100. Sin embargo, se ha observado que cuantos menos puntos se usen peores son las estimaciones de las matrices de proyección y esto es acusado por los algoritmos de autocalibración. Por eso y para no marear con demasiadas gráficas, sólo se presentan los resultados obtenidos con $n = 100$ puntos.

Una vez que se estima la matriz de parámetros intrínsecos $\hat{\mathtt{K}}$, los parámetros de calidad seleccionados para compararla con la matriz exacta $\bar{\mathtt{K}}$ son:

- **Error en la distancia focal**. Comparamos el error relativo

$$\varepsilon_\tau = \left| \frac{\bar{\alpha} - \hat{\alpha}}{\bar{\alpha}} \right| 100,$$

  medido en tanto por ciento.

- **Error en el punto principal**. Lo definimos como el cuadrado de la distancia euclídea en el plano de la imagen:

$$d^2 = (\bar{u}_0 - \hat{u}_0)^2 + (\bar{v}_0 - \hat{v}_0)^2$$

  Seguimos el mismo convenio que el que se utiliza para medir el rendimiento de los algoritmos cuyos costes son distancias geométricas. Si hallamos la raíz cuadrada de la media de las distancias al cuadrado, entonces es de esperar que el error siga una variación lineal con la desviación típica de ruido.

- **Error en la relación de aspecto**.Como se ha dicho, todos los experimentos generan matrices con píxeles cuadrados, así que mediremos el error relativo respecto del valor exacto $\bar{\tau} = 1$:

$$\varepsilon_\tau = \left| \frac{\bar{\tau} - \hat{\tau}}{\bar{\tau}} \right| 100.$$

- **Error en el ángulo entre los lados de los píxeles (skew)**. También utilizamos el error relativo, esta vez respecto del valor exacto $\bar{\theta} = \pi/2$:

$$\varepsilon_\theta = \left| \frac{\bar{\theta} - \hat{\theta}}{\bar{\theta}} \right| 100.$$

Las siglas mediante las cuales designaremos los algoritmos que comparamos son:

- CO: Cámaras ortogonales. Algoritmo lineal explicado en § 7.3.
- KRU: Ecuaciones de Kruppa, § 7.4.
- QLIN: algoritmo cuasi-lineal de Triggs de estimación de la cuádrica absoluta dual (ver § 7.6.1).
- QSQP: algoritmo original de Triggs, basado en la estimación de la cuádrica absoluta dual y optimización mediante la SQP e inicialización mediante el algoritmo lineal, ambos citados en § 7.6.1. Es el único que no se inicializa mediante el algoritmo CO.
- QCO: mismo algoritmo que el anterior, pero con la inicialización de cámaras ortogonales.

- QPV: algoritmo de estimación de la cuádrica absoluta dual mediante la minimización de la norma del producto vectorial de DIACs, ver § 7.6.2.
- QDE: algoritmo de estimación de la cuádrica absoluta dual mediante la minimización de la distancia esférica de DIACs. Explicado en § 7.6.3.
- QPVR: el mismo que el algoritmo QPV, pero imponiendo la restricción de píxeles cuadrados durante la optimización.
- QDER: igual que el algoritmo QDE, pero imponiendo la restricción de píxeles cuadrados durante la minimización.
- CHO: Estimación de la IAC mediante horópteras, § 7.8.5.
- QHO: Estimación de la cuádrica absoluta dual mediante horópteras, § 7.8.6.
- HA: algoritmo de Hartley, explicado en § 7.6.3.

Los algoritmos seleccionados son unos de los muchos posibles, mas el juego de combinaciones y comparativas es explosivo.

La entrada de los algoritmos de autocalibración son las matrices de proyección resultantes de una optimización mediante un ajuste de haces proyectivo. No sólo eso, sino que se multiplican por una matriz diagonal que realiza una normalización afín de las matrices sin cambiar la posición del punto principal de las cámaras ni la relación de aspecto ni el skew de los píxeles. La normalización afín habitual sí desplaza el punto principal.

En una simulación en la que el punto principal esté a ciencia cierta en el origen de coordenadas, interesa mantenerlo allí para utilizar como algoritmo de inicialización del resto de algoritmos el que utiliza la hipótesis de cámaras ortogonales. La matriz de la afinidad que realiza tal normalización se calcula a partir de los puntos ruidosos, no a partir de los puntos estimados tras el ajuste de haces.

Tanto la optimización de la calibración proyectiva como la nueva normalización afín (que equilibra los elementos de las matrices de proyección) persiguen el objetivo de mejorar el rendimiento de los algoritmos de autocalibración. Quizá se podría pensar en realizar la normalización afín al principio, antes de iniciar la calibración proyectiva. Algunos autores así lo sugieren. Sin embargo, no se ha hecho así porque al normalizar, se pierde el valor absoluto del error de reproyección: la escala no mide píxeles, sino "píxeles que han sufrido una transformación afín". Es lo que llamamos coste geométrico *normalizado* al hablar del algoritmo Gold Standard de estimación de la matriz de proyección.

Las figuras 7.8 y 7.9 presentan las curvas de los parámetros de error anteriormente definidos, para los experimentos con 20 mm de distancia focal, punto principal en el centro y píxeles cuadrados. Pasemos a comparar los distintos algoritmos entre sí.

En cuanto al error en la distancia focal, vemos que no superan la cota del 15 % para $\sigma = 5$. El algoritmo de las ecuaciones de Kruppa (KRU) ofrece resultados similares a los dos algoritmos lineales: CO y QLIN. Éstos son los que producen los mayores errores, seguidos del algoritmo QHO, que está a mitad camino entre el resto de algoritmos. Los algoritmos QSQP, QDE, QPV, QDER y QPVR dan buenas estimaciones (error del orden del 3 % para $\sigma = 5$ píxeles) y muy parecidas, ya que sus curvas de error son casi coincidentes. Los algoritmos QCO y CHO son todavía mejores y se acercan al 2 % de error.

En lo que al punto principal se refiere, en la gráfica derecha de la figura 7.8 está representada la raíz cuadrada del valor medio de las distancias al cuadrado, $\sqrt{E[\sum_i d_i^2]}$. Al ver la gráfica, se aprecia que es mucho suponer que se conoce la posición del punto principal en un algoritmo de autocalibración. A pesar de estar en una situación muy favorable, los errores medios no bajan de los 100 píxeles ($\sigma = 5$). El algoritmo QHO no se ha incluido en esta gráfica porque sus resultados están en clara desventaja

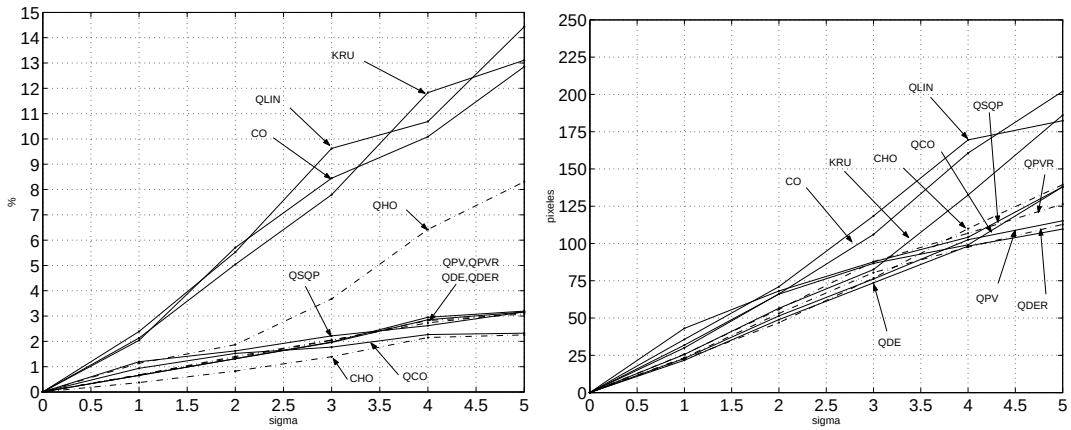


Figura 7.8: Autocalibración. Izquierda: errores en la distancia focal (20 mm). Derecha: errores en el punto principal (origen de coordenadas). En ambos, $n = 100$ puntos.

frente al resto. Podemos decir que es bueno estimando la distancia focal, pero malo estimando el resto de parámetros. Después, los algoritmos KRU, CO y QLIN son los que producen los mayores errores, aunque no son muy grandes en relación al resto: todos están por debajo de los 200 píxeles para $\sigma = 5$. Los algoritmos QSQP, QCO y CHO también producen estimaciones parecidas entre sí. Por último, los mejores respecto de este parámetro de calidad son los algoritmos QPVR, QPV, QDER y QDE.

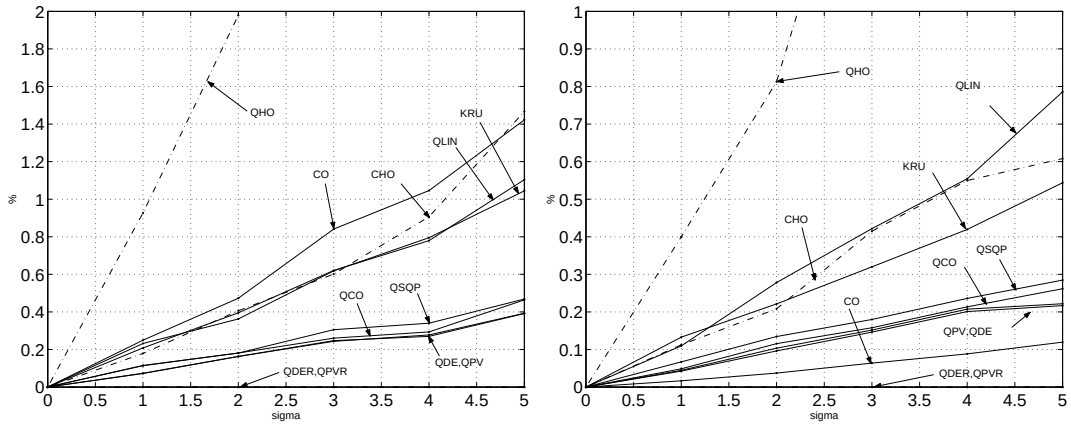


Figura 7.9: Autocalibración. Izquierda: errores en la relación de aspecto ($\tau = 1$). Derecha: errores en el skew ($\theta = \pi/2$), para $n = 100$ puntos.

En lo referente al error en la relación de aspecto y en el skew, la mayoría de los algoritmos producen buenos resultados, salvo el algoritmo QHO. Recordemos que estamos en la situación favorable de píxeles cuadrados. En general, los algoritmos QSQP, QCO, QDE y QPV son los que menores errores consiguen. Los algoritmos que imponen las restricciones de píxeles cuadrados, QPVR y QDER, tienen errores nulos en estos parámetros, como es lógico.

Las siguientes gráficas (figuras 7.10 y 7.11) muestran los experimentos realizados en las mismas condiciones que antes, pero cambiando la distancia focal a 30 mm y la distancia de la cámaras a la esfera de puntos.

Los errores en la estimación de la distancia focal y en el punto principal se han incrementado. La

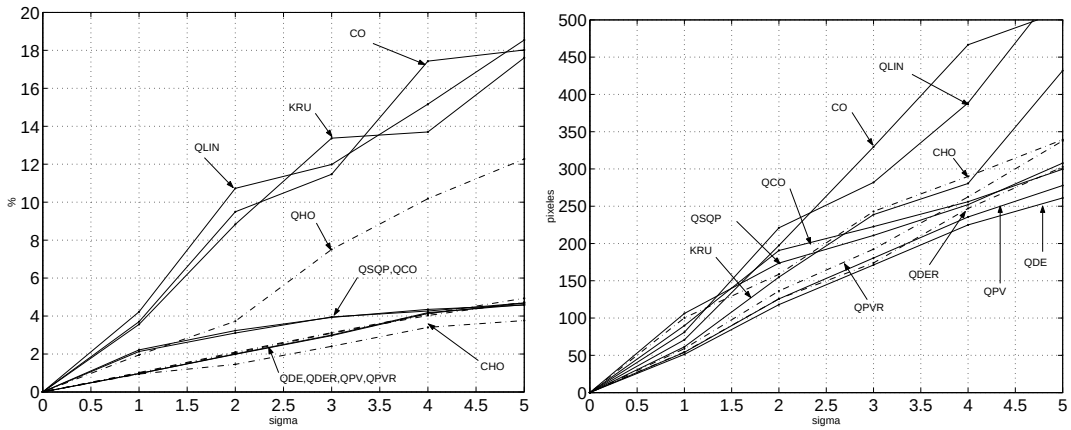


Figura 7.10: Autocalibración. Izquierda: errores en la distancia focal (30 mm). Derecha: errores en el punto principal (origen de coordenadas). En ambos, $n = 100$ puntos.

jerarquía de algoritmos se sigue cumpliendo: los dos algoritmos lineales (QLIN y CO) son los que mayores errores cometen, junto con el algoritmo de las ecuaciones de Kruppa (KRU). El siguiente en la escala es el algoritmo QHO, con un error del 12 % para $\sigma = 5$. En un error un poco superior al 4 % coinciden los algoritmos QSQP,QCO, QDE,QPV,QDER y QPVR. Los cuatro últimos siguen una variación más lineal que los dos que utilizan la SQP. Por último, El algoritmo CHO es el que mejor estima la distancia focal, con un error un poco inferior al 4 % para $\sigma = 5$.

Los puntos principales estimados difieren del real en más de 250 píxeles ($\sigma = 5$). Al igual que antes, los algoritmos KRU, CO y QLIN son los que más se equivocan, rondando los errores entre 400 y 500 píxeles. Por contra, los algoritmos QDE y QPV son los que menos se equivocan. En una zona intermedia están los algoritmos que imponen restricciones (QDER y QPVR) y los algoritmos basados en la SQP (QSQP y QCO).

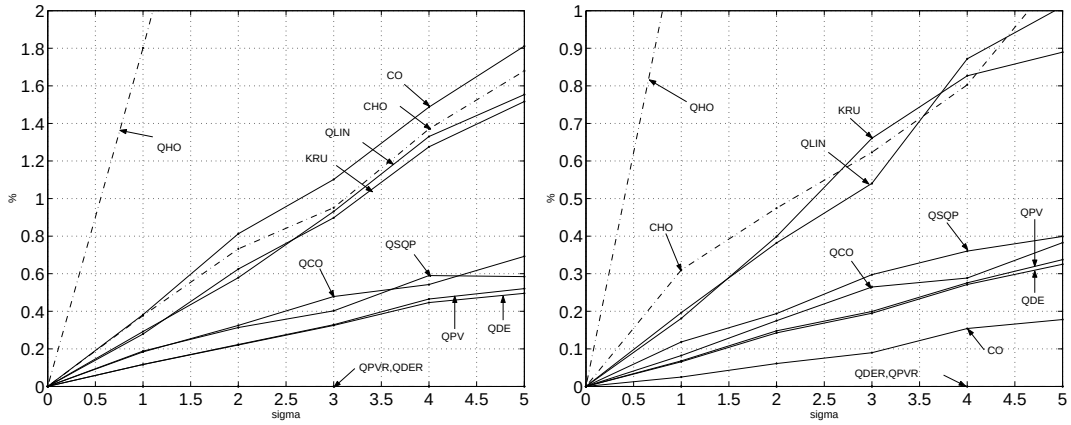


Figura 7.11: Autocalibración. Izquierda: errores en la relación de aspecto ($\tau = 1$). Derecha: errores en el skew ($\theta = \pi/2$). En ambos, $n = 100$ puntos.

Para los cuatro parámetros de error, las curvas de los algoritmos QDE, QDER, QPV y QPVR son las más lineales, lo que significa que los errores (en la distancia focal, el punto principal, etc.) son proporcionales a la desviación típica de ruido, σ.

Como se puede apreciar, los errores en la relación de aspecto y el skew siguen siendo pequeños, salvo para el algoritmo QHO. Al igual que antes, los algoritmos QPVR y QDER tienen un error nulo, ya que imponen las restricciones de píxeles cuadrados.

Si ahora variamos la distancia focal y movemos la posición del punto principal de la cámara (una vez más, la misma `K` para todas las imágenes), la inicialización mediante la suposición de cámaras ortogonales deja de ser eficiente. Los algoritmos no lineales producen mejores resultados si se utiliza previamente el algoritmo cuasi-lineal explicado en § 7.6.1, ya que éste no realiza ninguna suposición sobre la matriz de parámetros intrínsecos. El algoritmo de cámaras ortogonales (CO) están en desventaja respecto a éste (QLIN) si hay cuatro o más cámaras con los mismos parámetros intrínsecos. En cambio, hay que decir que el algoritmo CO da resultados con $m \geq 3$ cámaras, aunque tengan distintos parámetros intrínsecos.

Variamos los parámetros de la siguiente forma: la distancia focal tiene un valor medio de 20 mm ($\alpha_0 = 3{,}7796 \cdot 10^3$) y el valor medio del punto principal es el centro de la imagen. La distancia focal sigue una distribución uniforme en el intervalo $\alpha \in [\alpha_0 - \Delta\alpha, \alpha_0 + \Delta\alpha]$, con $\Delta\alpha = 0{,}1\alpha_0$. El punto principal también sigue una distribución uniforme, pero dentro del rectángulo de semilados $A = 2560/4$ y $B = 1920/4$ píxeles, centrado en el origen de coordenadas. Para cada experimento se elige una configuración de cámaras distinta, un conjunto de puntos 3D distintos y una matriz de parámetros intrínsecos distinta.

Las figuras 7.12 y 7.13 sintetizan los resultados de las pruebas realizadas con la anteriores modificaciones de los parámetros intrínsecos.

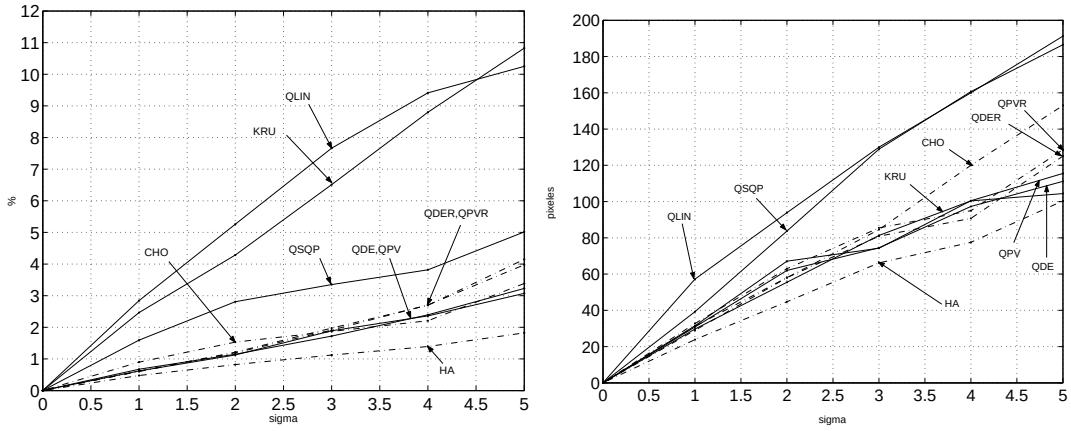


Figura 7.12: Autocalibración. Izquierda: errores en la distancia focal (valor medio: 20 mm). Derecha: errores en el punto principal (valor medio: origen de coordenadas). En ambos, $n = 100$ puntos.

El algoritmo CO no se encuentra en dichas gráficas, pero sus resultados son, a grandes rasgos: errores en la distancia focal superiores al 80 %, errores en el punto principal superiores a los 1000 píxeles, errores en la relación de aspecto comprendidos en el intervalo $2{,}5\,\% - 4\,\%$ y errores en el skew en el intervalo $0{,}85\,\% - 1{,}5\,\%$. Los resultados son malos a pesar de que no exista ruido en los datos, porque las hipótesis que maneja el algoritmo no se cumplen (punto principal en el centro). En cambio el algoritmo QLIN proporciona la solución exacta si el ruido es nulo: $\sigma = 0$, como recogen las gráficas.

De acuerdo con la explicación anterior, se ha eliminado de la comparativa los algoritmos CO y QCO. También se desecha el algoritmo QHO, puesto que también da malos resultados. En su lugar se ha incluido el algoritmo HA, que proporciona unos resultados excelentes.

A pesar de que la distancia focal varía en un $\pm 10\,\%$, la mayoría de los algoritmos se equivocan en menos de un $5\,\%$ para $\sigma = 5$, lo que son muy buenos resultados. El mejor, con diferencia, es el algoritmo HA, que se equivoca en menos de un $3\,\%$, mas los algoritmos QDE, QPV, QDER, QPVR y CHO también son muy buenos respecto de este parámetros de calidad.

Al comparar el error en el punto principal no se debe perder de vista las dimensiones del rectángulo donde puede variar. La mitad de la diagonal de dicho rectángulo vale $\sqrt{(2560/4)^2 + (1920/4)^2)} = 800$ píxeles, por lo que los errores que refleja la gráfica derecha de la figura 7.12 no son muy grandes: están comprendidos entre 100 y 200 píxeles para $\sigma = 5$. El rango de los errores es parecido al de la figura 7.8, a pesar de no estar el punto principal en el origen de coordenadas. Los mejores algoritmos son: HA, QDE, QPV y KRU, aunque éste último no sea bueno estimando la distancia focal.

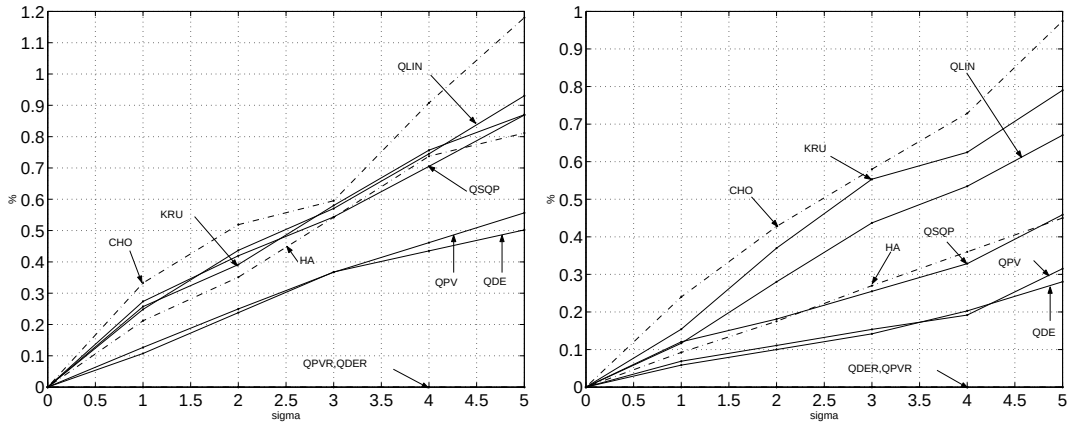


Figura 7.13: Autocalibración. Izquierda: errores en la relación de aspecto ($\tau = 1$). Derecha: errores en el skew ($\theta = \pi/2$). En ambos, $n = 100$ puntos.

Los errores en la relación de aspecto y en el skew siguen siendo pequeños, aproximadamente menores que el $1\,\%$. Los algoritmos QPVR y QDER mantienen el error nulo, como es de esperar.

En ninguna de las pruebas se ha medido el error en la matriz de rotación ni en la traslación que acompañan a la matriz de parámetros intrínsecos en la matriz de proyección ya que, aunque éstos errores sean pequeños, el error de reproyección de las matrices euclídeas correspondientes se dispara.

Las pruebas realizadas son muy costosas y podríamos seguir presentando más, ya que hay multitud de algoritmos implementados, sin embargo, éste no es el objetivo principal. Es más interesante pasar a la optimización euclídea, que seguro no es un tema habitual en la literatura.

Volviendo sobre las pruebas de este capítulo, no se ha hablado de los algoritmos que realizan la calibración estratificada comenzando por la restricción unimodular. Para más pruebas experimentales se pueden consultar las realizadas por los distintos inventores de los algoritmos en sus respectivos artículos.

Capítulo 8

Reconstrucción euclídea

El último bloque del esquema de procesamiento introducido en § 2.4 es el encargado de optimizar la reconstrucción euclídea que proporciona el algoritmo de autocalibración (posiciones de los puntos 3D y los parámetros de las matrices de proyección).

El capítulo está organizado en dos partes: la primera indica cómo obtener una reconstrucción euclídea inicial y la segunda trata la optimización de dicha reconstrucción mediante la técnica del ajuste de haces.

8.1. Reconstrucción inicial

Recordemos que para llegar a una reconstrucción euclídea hace falta identificar el plano del infinito π_∞ y la cónica absoluta Ω_∞ (o sus proyecciones ω o la matriz de parámetros intrínsecos de una cámara). Es posible obtener una reconstrucción euclídea a falta de una transformación de semejanza (rotación, traslación y escalado), para ser precisos deberíamos utilizar el término "reconstrucción métrica".

Supongamos que ya se ha realizado la autocalibración, entonces la la situación de partida está formada por las matrices de proyección modificadas por la homografía que transforma lo proyectivo en lo euclídeo, junto con los puntos 3D asociados. La construcción de esta homografía se explica en § 7.2.3. En realidad estas matrices todavía no son euclídeas: hay que aproximarlas por las euclídeas más cercanas. Hay dos formas de hacer esto:

1. Utilizar la descomposición QR para descomponer las tres primeras columnas de la matriz de proyección `P` en una matriz de rotación y una triangular superior.

2. Descomponer las tres primeras columnas de `P` en la matriz triangular superior dada por el algoritmo de autocalibración (`K` parámetros intrínsecos) y en la matriz de rotación que mejor se aproxima a la matriz.

Una vez que se sabe con certeza que las matrices son euclídeas, se puede pasar a optimizar la calibración euclídea.

8.2. Ajuste de Haces Euclídeo

La técnica habitual para refinar la solución obtenida tras la autocalibración es el ajuste de haces, que en esta caso llamaremos Ajuste de Haces Euclídeo, tema sobre el cual trata esta sección. No es complicado programar un ajuste de haces euclídeo con distorsión radial una vez que se conocen ambos elementos por separado, sin embargo no se ha realizado. Como observación, mencionaremos que el ajuste de haces se ha particularizado para el caso de cámaras con píxeles de forma conocida (τ y θ).

8.2.1. Parametrización de las matrices de proyección

Recordemos que una matriz de proyección métrica se puede descomponer de la forma

$$\mathtt{P}^j_{\mathrm{M}} = \mathtt{K}^j[\mathtt{R}^j \mid -\mathtt{R}^j\widetilde{\mathbf{C}}^j]$$

El primer paso consiste en obtener una parametrización de las matrices de proyección que permita optimizar sin salirse del espacio de soluciones que son matrices de proyección euclídeas.

Matrices de parámetros intrínsecos. En el caso particular de utilización del Calibration Pencil como algoritmo de autocalibración, se suponen conocidos la relación de aspecto $\tau = \alpha_u/\alpha_v$ y el ángulo θ entre los lados de los píxeles (skew). En las matrices de parámetros intrínsecos (ec. 4.2) debemos conservar estas restricciones y no optimizar respecto a estos parámetros, sólo respecto a los que faltan: la distancia focal codificada en α_v y el punto principal (u_0, v_0).

Se puede asumir, sin pérdida de generalidad que los píxeles son cuadrados ($\tau = 1$ y $\theta = \pi/2$), como se indica en § 7.9.2, y así se simplifican las operaciones y los cálculos de las derivadas, si proceden. La matriz de parámetros intrínsecos bajo esta suposición es:

$$\mathtt{K} = \begin{bmatrix} \alpha_v & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.1}$$

Matrices de rotación. Existen muchas formas de parametrizar las matrices de rotación (§ 5.2.1), cada una con sus ventajas e inconvenientes. Una matriz de rotación tiene 3 grados de libertad, por lo que el número mínimo de parámetros para construir una es 3. Después de probar varias opciones se ha decidido utilizar la matriz exponencial de una matriz antisimétrica como método de generación de la matriz de rotación. El vector $\mathbf{w}$ que define la matriz antisimétrica es el que parametriza la rotación:

$$\mathtt{R} = \exp([\mathbf{w}]_\times) = \exp([\hat{\omega}]_\times\ \theta)$$

Vectores de traslación. Los vectores de traslación o centros ópticos de las cámaras, $\widetilde{\mathbf{C}}$, están formados por las 3 coordenadas que definen la posición de un punto en $\mathbb{R}^3$, así que poseen otros 3 grados de libertad.

Recopilando todo lo anterior, bajo la suposición de píxeles cuadrados, cada matriz de proyección euclídea depende de 9 parámetros, $\mathbf{a}_j$.

$$\mathbf{a}_j = \begin{bmatrix} \alpha_v^j \\ u_0^j \\ v_0^j \\ \mathbf{w}^j \\ \widetilde{\mathbf{C}}^j \end{bmatrix} \longrightarrow \begin{bmatrix} \mathtt{K}^j \\ \mathtt{R}^j \\ \widetilde{\mathbf{C}}^j \end{bmatrix} \longrightarrow \mathtt{P}^j_{\mathrm{M}}$$

Toda esta discusión de la parametrización también está justificada porque hay que proporcionar al algoritmo de optimización (Levenberg-Marquardt) un punto de partida de la búsqueda. Ya se dispone de las coordenadas afines de los puntos 3D del vector $\mathbf{P}$, falta a partir de cada matriz de proyección rectificada obtener los parámetros $\mathbf{a}^j$ que la aproximan. De las dos opciones explicadas en § 8.1 sólo ha sido probada la primera opción y funciona.

8.2.2. Funciones modelo y de coste

Respecto del ajuste de haces proyectivo, § 6.6.4, cambia la descripción de las matrices de proyección, es decir, la estructura de la parte $\mathbf{a}$ del vector de parámetros $\mathbf{P}$ en el marco común de los algoritmos de Levenberg-Marquardt. Ahora, la dimensión de $\mathbf{P}$ es $M = 9m + 3n$, en lugar de $M = 12m + 3n$.

También cambia la matriz jacobiana, de dimensiones $\dim(\widehat{\mathbf{X}}) \times \dim(\mathbf{P}) = (2mn) \times (9m + 3n)$.

Veamos las funciones que describen el problema en el marco común de aplicación del algoritmo LM. Función modelo:

$$f : \left[\begin{array}{c}\mathbf{a}\\ \hline \mathbf{b}\end{array}\right] = \left[\begin{array}{c}\mathbf{a}_1\\ \vdots\\ \mathbf{a}_m\\ \hline \mathbf{b}_1\\ \vdots\\ \mathbf{b}_n\end{array}\right] = \left[\begin{array}{c}\mathbf{a}_1\\ \vdots\\ \mathbf{a}_m\\ \hline \hat{\mathbf{X}}_{\mathrm{M}\,a1}\\ \vdots\\ \hat{\mathbf{X}}_{\mathrm{M}\,a\,n}\end{array}\right] \equiv \mathbf{P} \to \widehat{\mathbf{X}} \equiv \left[\begin{array}{c}\left(\begin{array}{c}\hat{\mathbf{x}}^1_{a1}\\ \vdots\\ \hat{\mathbf{x}}^m_{a1}\end{array}\right)\\ \vdots\\ \left(\begin{array}{c}\hat{\mathbf{x}}^1_{a\,n}\\ \vdots\\ \hat{\mathbf{x}}^m_{a\,n}\end{array}\right)\end{array}\right] = \left[\begin{array}{c}\widehat{\mathbf{X}}_{a1}\\ \vdots\\ \widehat{\mathbf{X}}_{a\,n}\end{array}\right]$$

El vector de parámetros $\mathbf{P}$, está dividido en dos partes: los parámetros de las matrices de proyección métricas y los parámetros de los puntos 3D. El vector de medidas se construye con las coordenadas afines de los puntos proyectados y pertenece a un espacio de dimensión $N = 2mn$. La función de coste es el error de reproyección de los n puntos en las m imágenes (distancia geométrica).

$$\text{coste} = \|\mathbf{X} - \widehat{\mathbf{X}}\| = \|\mathbf{X} - f(\mathbf{P})\| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} d(\hat{\mathtt{P}}^i\hat{\mathbf{X}}_j, \mathbf{x}^i_j)^2}$$

En estos algoritmos de grandes dimensiones se recomienda calcular la matriz jacobiana de forma exacta, ya que es mucho más rápida que la evaluación numérica de la misma (requiere muchas menos evaluaciones de la función). La obtención de dicha matriz se explica a continuación.

8.2.3. Interpretación: composición de funciones

Para aplicar el algoritmo LM con rapidez, necesitamos conocer la matriz jacobiana $\mathtt{J}_f$ de la función modelo del ajuste de haces euclídeo f (llamada así en § 8.2.2). Lo mejor (ejecución más rápida y precisa) es proporcionar las derivadas exactas, en lugar de estimarlas numéricamente. Al calcular dichas derivadas se obtienen unas expresiones muy complicadas (obtenidas simbólicamente, con Maple), problema que resolvemos en esta sección.

El primer paso hacia la resolución consiste en darse cuenta de que la función f la podemos expresar como composición de dos funciones, de las cuales una de ellas, h es familiar y se conoce la expresión de su matriz jacobiana.

$$f = h \circ g \tag{8.2}$$

La función h se corresponde con la función modelo del ajuste de haces proyectivo (§ 6.6.4). Y la función g es la que transforma los parámetros de las matrices euclídeas en los elementos de dichas matrices. Esto se recoge en el siguiente diagrama.

$$\begin{array}{ccccc}
\mathbb{R}^L & \xrightarrow{\quad g \quad} & \mathbb{R}^M & \xrightarrow{\quad h \quad} & \mathbb{R}^N\\
\mathbf{P} & & \mathbf{Z} & & \mathbf{X}\\
\left[\begin{array}{c}\mathbf{a}_1\\ \vdots\\ \mathbf{a}_m\\ \hline \hat{\mathbf{X}}_{\mathrm{M}\,a1}\\ \vdots\\ \hat{\mathbf{X}}_{\mathrm{M}\,a\,n}\end{array}\right] & & \left[\begin{array}{c}\mathtt{P}^1_{\mathrm{M}}\\ \vdots\\ \mathtt{P}^m_{\mathrm{M}}\\ \hline \hat{\mathbf{X}}_{\mathrm{M}\,a1}\\ \vdots\\ \hat{\mathbf{X}}_{\mathrm{M}\,a\,n}\end{array}\right] & & \left[\begin{array}{c}\left(\begin{array}{c}\hat{\mathbf{x}}^1_{a1}\\ \vdots\\ \hat{\mathbf{x}}^m_{a1}\end{array}\right)\\ \vdots\\ \left(\begin{array}{c}\hat{\mathbf{x}}^1_{a\,n}\\ \vdots\\ \hat{\mathbf{x}}^m_{a\,n}\end{array}\right)\end{array}\right]
\end{array}$$

En el caso considerado, las dimensiones de los diferentes espacios son

$$\begin{aligned} L &= 9m + 3n \\ M &= 12m + 3n \\ N &= 2mn \end{aligned}$$

La ventaja de expresar f como composición de estas dos funciones es que podemos aplicar la regla de la cadena y concluir que la matriz jacobiana de la composición de funciones es el producto de las matrices jacobianas de cada función individual.

$$\begin{array}{ccccc} \mathbb{R}^L & \xrightarrow{\quad g \quad} & \mathbb{R}^M & \xrightarrow{\quad h \quad} & \mathbb{R}^N \\ \mathbf{P} & & \mathbf{Z} & & \mathbf{X} \\ & \mathtt{J}_g = \left[\dfrac{\partial \mathbf{Z}}{\partial \mathbf{P}}\right] & & \mathtt{J}_h = \left[\dfrac{\partial \mathbf{X}}{\partial \mathbf{Z}}\right] & \\ & M \times L & & N \times M & \end{array}$$

Por la regla de la cadena:

$$\begin{aligned} \mathtt{J}_f &= \mathtt{J}_h \mathtt{J}_g \\ \left[\frac{\partial \mathbf{X}}{\partial \mathbf{P}}\right] &= \left[\frac{\partial \mathbf{X}}{\partial \mathbf{Z}}\right]\left[\frac{\partial \mathbf{Z}}{\partial \mathbf{P}}\right] \\ N \times L &= (N \times M)(M \times L) \end{aligned} \tag{8.3}$$

Para el ajuste de haces, todas las matrices jacobianas consideradas son dispersas. La matriz $\mathtt{J}_g$ es diagonal a bloques. De hecho, la función g sólo se encarga de obtener las matrices de proyección a partir de sus parámetros, no modifica las coordenadas afines de los puntos 3D, por lo que la parte de la $\mathtt{J}_g$ correspondiente a dichos puntos es la identidad.

$$\mathtt{J}_g = \left[\begin{array}{c|c} \begin{array}{ccc} \mathtt{J}_g^1 & & \\ & \mathtt{J}_g^2 & \\ & & \ddots \\ & & \mathtt{J}_g^m \end{array} & \begin{array}{ccc} & 0 & \\ \mathtt{I}_3 & & \\ & \mathtt{I}_3 & \\ 0 & & \ddots \\ & & \mathtt{I}_3 \end{array} \end{array}\right] = \left[\begin{array}{c|c} \mathrm{diag}(\mathtt{J}_g^i) & 0 \\ \hline 0 & \mathtt{I} \end{array}\right]$$

En nuestro caso, debido a la parametrización elegida para las matrices de proyección euclídeas (§ 8.2.1), cada matriz $\mathtt{J}_g^i$ tiene dimensiones 12×9. El número de columnas es el número de variables independientes de la cámara: de los 11 grados de libertad se restan 2 al considerar píxeles cuadrados.

Además, consideremos la matriz $\mathtt{J}_h$, cuya estructura de matriz dispersa en términos de las submatrices

jacobianas $\mathtt{A}_{ij}$ y $\mathtt{B}_{ij}$ es la siguiente:

$$\mathtt{J}_h = \left[\begin{array}{cccc|ccc} \mathtt{A}_{11} & & & & \mathtt{B}_{11} & & \\ & \mathtt{A}_{12} & & & \mathtt{B}_{12} & & \\ & & \ddots & & \vdots & & \\ & & & \mathtt{A}_{1m} & \mathtt{B}_{1m} & & \\ \mathtt{A}_{21} & & & & & \mathtt{B}_{21} & \\ & \mathtt{A}_{22} & & & & \mathtt{B}_{22} & \\ & & \ddots & & & \vdots & \\ & & & \mathtt{A}_{2m} & & \mathtt{B}_{2m} & \\ \mathtt{A}_{n\,1} & & & & & & \mathtt{B}_{n\,1} \\ & \mathtt{A}_{n\,2} & & & & & \mathtt{B}_{n\,2} \\ & & \ddots & & & & \vdots \\ & & & \mathtt{A}_{n\,m} & & & \mathtt{B}_{n\,m} \end{array}\right]$$

La matriz $\mathtt{J}_f$, empleando la regla de la cadena (8.3), posee la misma estructura dispersa que $\mathtt{J}_h$, pero cada submatriz $\mathtt{A}'_{ij} = \mathtt{A}_{ij}\mathtt{J}_g^j$ es de 2×9, en lugar de 2×12, según la parametrización elegida. Las submatrices $\mathtt{B}_{ij}$ no cambian ya que la función g no modifica los puntos 3D.

$$\mathtt{J}_h = \left[\begin{array}{cccc|ccc} \mathtt{A}_{11}\mathtt{J}_g^1 & & & & \mathtt{B}_{11} & & \\ & \mathtt{A}_{12}\mathtt{J}_g^2 & & & \mathtt{B}_{12} & & \\ & & \ddots & & \vdots & & \\ & & & \mathtt{A}_{1m}\mathtt{J}_g^m & \mathtt{B}_{1m} & & \\ \mathtt{A}_{21}\mathtt{J}_g^1 & & & & & \mathtt{B}_{21} & \\ & \mathtt{A}_{22}\mathtt{J}_g^2 & & & & \mathtt{B}_{22} & \\ & & \ddots & & & \vdots & \\ & & & \mathtt{A}_{2m}\mathtt{J}_g^m & & \mathtt{B}_{2m} & \\ \mathtt{A}_{n\,1}\mathtt{J}_g^1 & & & & & & \mathtt{B}_{n\,1} \\ & \mathtt{A}_{n\,2}\mathtt{J}_g^2 & & & & & \mathtt{B}_{n\,2} \\ & & \ddots & & & & \vdots \\ & & & \mathtt{A}_{n\,m}\mathtt{J}_g^m & & & \mathtt{B}_{n\,m} \end{array}\right]$$

En resumen:

$$\left.\begin{array}{l} \mathtt{J}_g = \left\{\begin{array}{l} \mathtt{J}_g^i \\ \mathtt{I}_3 \end{array}\right. \\ \mathtt{J}_h = \left\{\begin{array}{l} \mathtt{A}_{ij} \\ \mathtt{B}_{ij} \end{array}\right. \end{array}\right\} \longrightarrow \quad \mathtt{J}_f = \left\{\begin{array}{lcl} \mathtt{A}'_{ij} & = & \mathtt{A}_{ij}\mathtt{J}_g^j \\ \mathtt{B}'_{ij} & = & \mathtt{B}_{ij}\mathtt{I}_3 = \mathtt{B}_{ij} \end{array}\right.$$

Toda la información de la variación de las matrices de proyección según sus parametrizaciones (contenida en la matriz $\mathtt{J}_g$) queda reflejada en las matrices $\mathtt{J}_g^i$, de las cuales sólo hay m distintas, no hace falta calcular más, aunque haya mn matrices $\mathtt{A}_{ij}$ distintas. Veamos a continuación cómo calcular una de las matrices jacobianas $\mathtt{J}_g^i$.

8.2.3.1. Derivadas respecto de los parámetros euclídeos

Supongamos que se elige una cámara euclídea cualquiera $\mathtt{P}_{\mathrm{M}}^i = \mathtt{K}^i\mathtt{R}^i[\mathtt{I} \mid -\widetilde{\mathbf{C}}^i]$ y se desea conocer la matriz jacobiana $\mathtt{J}_g^i$. Escribamos la matriz de proyección euclídea en función de cada elemento de $\mathtt{K}$, $\mathtt{R}$ y $\widetilde{\mathbf{C}} = (t_1, t_2, t_3)^\top$.

$$\mathtt{P} = \left[\begin{array}{cccc} \alpha_v R_{11}+u_0 R_{31} & \alpha_v R_{12}+u_0 R_{32} & \alpha_v R_{13}+u_0 R_{33} & -[(\alpha_v R_{11}+u_0 R_{31})t_1+(\alpha_v R_{12}+u_0 R_{32})t_2+(\alpha_v R_{13}+u_0 R_{33})t_3] \\ \alpha_v R_{21}+v_0 R_{31} & \alpha_v R_{22}+v_0 R_{32} & \alpha_v R_{23}+v_0 R_{33} & -[(\alpha_v R_{21}+v_0 R_{31})t_1+(\alpha_v R_{22}+v_0 R_{32})t_2+(\alpha_v R_{23}+v_0 R_{33})t_3] \\ R_{31} & R_{32} & R_{33} & -(R_{31}t_1+R_{32}t_2+R_{33}t_3) \end{array}\right]$$

La matriz jacobiana $\mathtt{J}_g^i$ de los elementos de una matriz de proyección euclídea p_{ij} respecto de los parámetros que la definen $\mathbf{a} = (\alpha_v, u_0, v_0, \mathbf{w}, \widetilde{\mathbf{C}})$ es de tamaño 12×9 y se puede dividir en tres submatrices:

$$\mathtt{J}_g = \begin{bmatrix} \mathtt{J}_\mathtt{K} & \mathtt{J}_\mathtt{R} & \mathtt{J}_{\widetilde{\mathbf{C}}} \end{bmatrix}$$

- Derivadas de de la matriz de proyección $\mathtt{P}$ respecto de los parámetros intrínsecos:

$$\mathtt{J}_\mathtt{K} = \begin{array}{c} \\ p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{array} \begin{array}{c} \begin{array}{ccc} \alpha_v & u_0 & v_0 \end{array} \\ \begin{pmatrix} R_{11} & R_{31} & 0 \\ R_{12} & R_{32} & 0 \\ R_{13} & R_{33} & 0 \\ -\mathbf{r}_{1\star}^\top \widetilde{\mathbf{C}} & -\mathbf{r}_{3\star}^\top \widetilde{\mathbf{C}} & 0 \\ R_{21} & 0 & R_{31} \\ R_{22} & 0 & R_{32} \\ R_{23} & 0 & R_{33} \\ -\mathbf{r}_{2\star}^\top \widetilde{\mathbf{C}} & 0 & -\mathbf{r}_{3\star}^\top \widetilde{\mathbf{C}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \end{array} = \begin{pmatrix} \mathbf{r}_{1\star} & \mathbf{r}_{3\star} & \mathbf{0}_3 \\ -\mathbf{r}_{1\star}^\top \widetilde{\mathbf{C}} & -\mathbf{r}_{3\star}^\top \widetilde{\mathbf{C}} & 0 \\ \mathbf{r}_{2\star} & \mathbf{0}_3 & \mathbf{r}_{3\star} \\ -\mathbf{r}_{2\star}^\top \widetilde{\mathbf{C}} & 0 & -\mathbf{r}_{3\star}^\top \widetilde{\mathbf{C}} \\ \mathbf{0}_4 & \mathbf{0}_4 & \mathbf{0}_4 \end{pmatrix}$$

  siendo $\mathbf{r}_{i\star}^\top = (R_{i1}, R_{i2}, R_{i3})$ la fila i-esima de $\mathtt{R}$. La estrella $\star$ se sitúa en el índice donde va la columna, para mantener notación de subíndices.

- Derivadas de la matriz de proyección $\mathtt{P}$ respecto de la rotación.

  Notemos las derivadas parciales de los elementos de la matriz de rotación respecto de el vector $\mathbf{w} = (w_1, w_2, w_3)^\top$ que la parametriza mediante la siguiente expresión:

$$R_{mn}^k = \frac{\partial R_{mn}}{\partial w_k}$$

  Sea

$$\mathrm{d}\mathtt{R}^k = (R_{11}^k, R_{12}^k, R_{13}^k, R_{21}^k, R_{22}^k, R_{23}^k, R_{31}^k, R_{32}^k, R_{33}^k)^\top$$

  Así, obtenemos la siguiente matriz 9×3 de derivadas parciales.

$$\mathrm{d}\mathtt{R} = \begin{bmatrix} \mathrm{d}\mathtt{R}^1 & \mathrm{d}\mathtt{R}^2 & \mathrm{d}\mathtt{R}^3 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_{1\star}^1 & \mathbf{r}_{1\star}^2 & \mathbf{r}_{1\star}^3 \\ \mathbf{r}_{2\star}^1 & \mathbf{r}_{2\star}^2 & \mathbf{r}_{2\star}^3 \\ \mathbf{r}_{3\star}^1 & \mathbf{r}_{3\star}^2 & \mathbf{r}_{3\star}^3 \end{bmatrix}$$

  siendo $\mathbf{r}_{i\star}^k = (R_{i1}^k, R_{i2}^k, R_{i3}^k)^\top$ (i indica la fila de $\mathtt{R}$ y k señala la componente del vector $\mathbf{w}$). Es fácil expresar la submatriz $\mathtt{J}_\mathtt{R}$ en función de esta última matriz.

$$\mathtt{J}_\mathtt{R} = \begin{bmatrix} \alpha_v \mathbf{r}_{1\star}^1 + u_0 \mathbf{r}_{3\star}^1 & \alpha_v \mathbf{r}_{1\star}^2 + u_0 \mathbf{r}_{3\star}^2 & \alpha_v \mathbf{r}_{1\star}^3 + u_0 \mathbf{r}_{3\star}^3 \\ -\widetilde{\mathbf{C}}^\top \mathtt{J}_\mathtt{R}(1:3,1) & -\widetilde{\mathbf{C}}^\top \mathtt{J}_\mathtt{R}(1:3,2) & -\widetilde{\mathbf{C}}^\top \mathtt{J}_\mathtt{R}(1:3,3) \\ \alpha_v \mathbf{r}_{2\star}^1 + v_0 \mathbf{r}_{3\star}^1 & \alpha_v \mathbf{r}_{2\star}^2 + v_0 \mathbf{r}_{3\star}^2 & \alpha_v \mathbf{r}_{2\star}^3 + v_0 \mathbf{r}_{3\star}^3 \\ -\widetilde{\mathbf{C}}^\top \mathtt{J}_\mathtt{R}(5:7,1) & -\widetilde{\mathbf{C}}^\top \mathtt{J}_\mathtt{R}(5:7,2) & -\widetilde{\mathbf{C}}^\top \mathtt{J}_\mathtt{R}(5:7,3) \\ \mathbf{r}_{3\star}^1 & \mathbf{r}_{3\star}^2 & \mathbf{r}_{3\star}^3 \\ -\widetilde{\mathbf{C}}^\top \mathtt{J}_\mathtt{R}(9:11,1) & -\widetilde{\mathbf{C}}^\top \mathtt{J}_\mathtt{R}(9:11,2) & -\widetilde{\mathbf{C}}^\top \mathtt{J}_\mathtt{R}(9:11,3) \end{bmatrix}$$

  En esta última expresión ha sido utilizada la notación de MATLAB ":" para matrices, por simplicidad de escritura.

Para no marear la argumentación con engorrosas fórmulas se ha dejado para el final de este apartado la obtención explícita de dR (ya se apreciará el porqué).

Fijémonos en la fórmula de Rodrigues $\mathtt{R} = \exp([\mathbf{w}]_\times)$, y sea el ángulo $\theta = \|\mathbf{w}\|$ la norma de $\mathbf{w}$. Nótese que $\theta = \theta(w_1, w_2, w_3)$, no se añaden más parámetros a los ya existentes. La matriz de la rotación es la siguiente:

$$\begin{bmatrix} \cos\theta + \dfrac{w_1^2(1-\cos\theta)}{\theta^2} & \dfrac{w_1w_2(1-\cos\theta) - \theta w_3 \operatorname{sen}\theta}{\theta^2} & \dfrac{\theta w_2 \operatorname{sen}\theta + w_1w_3(1-\cos\theta)}{\theta^2} \\ \dfrac{\theta w_3 \operatorname{sen}\theta + w_1w_2(1-\cos\theta)}{\theta^2} & \cos\theta + \dfrac{w_2^2(1-\cos\theta)}{\theta^2} & \dfrac{-\theta w_1 \operatorname{sen}\theta + w_2w_3(1-\cos\theta)}{\theta^2} \\ \dfrac{-\theta w_2 \operatorname{sen}\theta + w_1w_3(1-\cos\theta)}{\theta^2} & \dfrac{\theta w_1 \operatorname{sen}\theta + w_2w_3(1-\cos\theta)}{\theta^2} & \cos\theta + \dfrac{w_3^2(1-\cos\theta)}{\theta^2} \end{bmatrix}$$

(Obviamente, si $\theta = 0$, la matriz es la identidad).

Si por ejemplo, derivamos el elemento R_{11} respecto de w_1, queda una fórmula no trivial a simple vista:

$$\frac{\partial R_{11}}{\partial w_1} = -w_1 \frac{2(\theta^2 + w_1^2)(1-\cos\theta) + (\theta^2 - w_1^2)\theta \sin\theta}{\theta^4}.$$

Lo mismo sucede con el resto de derivadas parciales. La mejor solución es calcular las expresiones simbólicamente (con Maple) y luego incluirlas en un fichero en MATLAB que devuelva dR para una matriz de rotación dada, R.

Sólo hay que tener cuidado en caso de ver la variación si la rotación es la identidad (condición que se suele utilizar para la primera cámara). En esta situación se debe hallar el límite de las expresiones R_{ij}^k cuando $w_k \to 0$ y las otras componentes de $\mathbf{w}$ se anulan. Dichos límites salen finitos, a pesar de la indeterminación 0/0 típica de las escondidas sincs que hay en las fórmulas. Esta es la forma de tratar los avisos de división por cero que pudieran aparecer si no se llevara cuidado. Para la rotación identidad, la matriz de derivadas parciales dR es:

$$\mathtt{dR} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

- Derivadas de de la matriz de proyección P respecto de la traslación son muy fáciles:

$$\mathtt{J}_{\widetilde{\mathbf{C}}} = \begin{array}{c} \\ p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{array} \begin{array}{c} \begin{array}{ccc} t_1 & t_2 & t_3 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -(\alpha_v R_{11} + u_0 R_{31}) & -(\alpha_v R_{12} + u_0 R_{32}) & -(\alpha_v R_{13} + u_0 R_{33}) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -(\alpha_v R_{21} + v_0 R_{31}) & -(\alpha_v R_{22} + v_0 R_{32}) & -(\alpha_v R_{23} + v_0 R_{33}) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -R_{31} & -R_{32} & -R_{33} \end{pmatrix} \end{array}$$

8.3. Evaluación experimental

Para probar el rendimiento del ajuste de haces euclídeo se han hecho experimentos completos de calibración: partiendo de los datos sintéticos ruidosos, se realiza una calibración proyectiva optimizada mediante el ajuste de haces. A continuación se obtienen los elementos de autocalibración y se construye la homografía para convertir la reconstrucción proyectiva en una euclídea. Por último se optimiza la calibración euclídea.

El número de cámaras elegido es $m = 5$, y el número de puntos es variable. Debido a que los algoritmos de autocalibración son muy sensibles al ruido, se utiliza un número de puntos comprendido entre 50 y 100, suficientemente grande para que las matrices de proyección produzcan buenos resultados de calibración. Las cámaras poseen todas la misma matriz de parámetros intrínsecos, desconocida, aunque en las pruebas todas poseen píxeles cuadrados. Los parámetros tienen los siguientes valores medios: 20 mm de distancia focal ($\alpha_0 = 3{,}7796 \cdot 10^3$) y punto principal situado en el centro de la imagen. La distancia focal varía uniformemente dentro de un margen del $\pm 10\,\%$ ($\Delta\alpha = 0{,}1\alpha_0$), es decir, $\alpha \in [\alpha_0 - \Delta\alpha, \alpha_0 + \Delta\alpha]$ mientras que el punto principal sigue una distribución uniforme dentro del rectángulo de semilados $A = 2560/4$ y $B = 1920/4$ píxeles, centrado en el origen de coordenadas. El número de experimentos realizados es 100, suficiente para que las medias sean significativas.

El algoritmo de autocalibración elegido es el de estimación de la cuádrica absoluta dual mediante la técnica de programación cuadrática secuencial (SQP), propuesto por Triggs (ver § 7.6.1). El algoritmo no impone restricciones sobre los parámetros: aunque en la simulación se conozca que los píxeles son cuadrados, no se utiliza en la determinación de las matrices de parámetros intrínsecos. Este algoritmo se inicializa mediante el algoritmo cuasi-lineal del mismo autor [15], que proporciona un mejor punto de partida para la optimización no lineal que el algoritmo de cámaras ortogonales.

A continuación del algoritmo de autocalibración se construye la homografía `H` que actualiza la reconstrucción proyectiva en una euclídea (ver § 7.2.3) y se prueban las dos estrategias de inicialización del ajuste de haces euclídeo presentadas en § 8.1. La mejor alternativa es la primera.

Recordemos que en la segunda opción se impone que la submatriz formada por las tres primeras columnas de la matriz de proyección `PH` se obtenga como producto de la matriz de parámetros intrínsecos que devuelve el algoritmo de autocalibración por la matriz de rotación más cercana al resto de la descomposición. En la práctica, debido al ruido, la matriz de parámetros intrínsecos estimada $\hat{\mathtt{K}}$ difiere (sobre todo en lo que a la estimación del punto principal se refiere) de la matriz exacta `K`, por lo que al intentar descomponer la matriz de proyección con esa matriz $\hat{\mathtt{K}}$, el resultado es que no encaja bien. Esto se observa en que el error de reproyección de los puntos 3D a través de las matrices euclídeas toma valores muy grandes comparados con los que cabría esperar debido al ajuste de haces proyectivo realizado.

En la primera alternativa, se descompone cada submatriz de la matriz de proyección en una matriz de rotación y una matriz de parámetros intrínsecos (la que sea — no tiene porqué coincidir con la estimada en la autocalibración, salvo para la primera cámara, debido a la homografía `H`). Esto es mucho mejor, ya que tiene mayor libertad: cada matriz de proyección se ajusta por separado.

Con pruebas experimentales se ha comprobado esta afirmación de que la primera inicialización de § 8.1 es mejor que la segunda. En las pruebas no se ha realizado la normalización afín de los puntos ruidosos para dar un significado de píxeles (absolutos) a los errores (distancias geométricas) obtenidos en cada paso de la cadena de procesamiento. Cabe todavía la esperanza de que la segunda inicialización funcione mejor si se trabaja en un mundo de coordenadas normalizadas (afínmente), ya que la rotación más próxima que se ajusta a la matriz $\hat{\mathtt{K}}$ y a la matriz de proyección actualizada, `PH`, minimiza la norma matricial inducida, que al igual que se estudió en el caso de la matriz fundamental, funciona mejor si se trabaja de tal forma que las coordenadas de los puntos (y por tanto los elementos de las matrices de proyección) estén equilibradas.

Tras la anterior justificación, se procede a evaluar experimentalmente sólo la primera inicialización. Las gráficas de la figura 8.1 muestran los resultados de las pruebas. En ellas se comparan cuatro resultados de la cadena de procesamiento (ver § 2.4): la reconstrucción proyectiva inicial (INIC−CP) obtenida, como se explicó en § 6.6.3, mediante la matriz fundamental y resección; el resultado del ajuste de haces proyectivo (AHP); la reconstrucción euclídea inicial (INIC−CE) imponiendo la restricción de píxeles cuadrados y el resultado del ajuste de haces euclídeo (AHE).

Como es habitual, las líneas discontinuas representan el error de estimación (distancia de los puntos estimados a los puntos exactos) y las líneas continuas representan el error residual (distancia a los datos ruidosos).

Como se puede comprobar en la figura, la recta de la reconstrucción euclídea inicial más píxeles cuadrados (INIC−CE) tiene una pendiente considerablemente mayor que la del resto de calibraciones. Los errores son mayores cuantos menos puntos se utilicen para estimar las matrices de proyección. Si sólo se ajustasen las matrices de proyección a unas de cámaras proyectivas finitas (sin imponer que los píxeles sean cuadrados), la curva del coste de la reconstrucción inicial coincidiría con la del ajuste de haces proyectivo, aproximadamente.

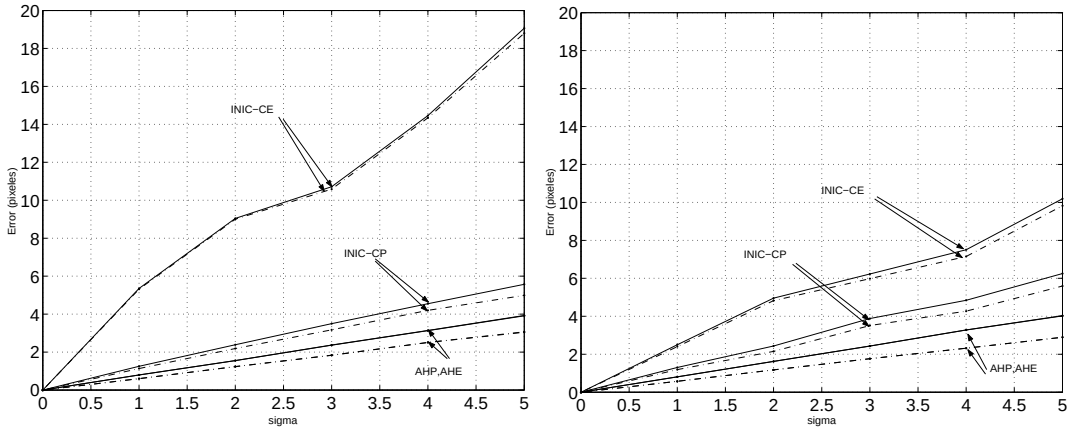


Figura 8.1: Ajuste de haces euclídeo: valores RMS residual y de estimación del error de reproyección, para $n = 50$ puntos (izquierda) y $n = 100$ (derecha).

El ajuste de haces euclídeo en verdad es ventajoso, ya que optimiza la calibración euclídea y consigue que el error de reproyección se mantenga cerca de los valores mínimos teóricos. Prácticamente, la curva AHE coincide con la curva del ajuste de de haces proyectivo, AHP. Esto era de esperar, pues del análisis teórico realizado en § 6.7.5 sabemos que las superficies de error proyectivas y métricas son asintóticamente convergentes según crecen el número de puntos n y el número de cámaras m.

En ambas gráficas se observan las dos predicciones de las curvas teóricas (ver figura 6.31): (1) conforme crece el número de puntos, el error residual aumenta y el error de estimación disminuye; (2) la variación de ambos errores de reproyección con la desviación típica de ruido es lineal.

Queda un detalle por comentar: el ajuste de haces diseñado no impone que todas las matrices de parámetros intrínsecos sean iguales. Está pensado para el caso de cámaras con píxeles cuadrados y parámetros intrínsecos variables, aunque lo hallamos utilizado para cámaras con parámetros fijos. El algoritmo funciona igual de bien si se utilizan cámaras con parámetros variables.

Como datos complementarios sobre el algoritmo de autocalibración, para $n = 100$ puntos los errores

en la distancia focal se mantienen por debajo del 4,5 % (para $\sigma = 5$ píxeles), el error medio en el punto principal está por debajo de 200 píxeles, el error en la relación de aspecto es inferior al 0,5 % y el error en el ángulo del skew es también inferior al 0,5 %. Para $n = 50$ puntos, el error en la estimación de la distancia focal es menor que el 7,5 %, el error en el punto principal es inferior a 250 píxeles y los errores en la relación de aspecto τ y ángulo θ entre los lados de los píxeles son inferiores al 1 % y al 0,8 %, respectivamente.

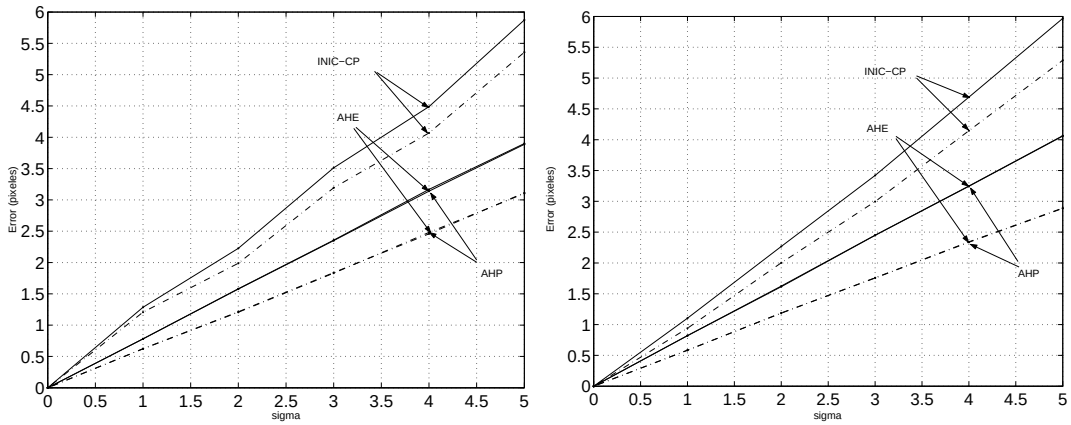


Figura 8.2: Ajuste de haces euclídeo: detalle de los valores RMS residual y de estimación del error de reproyección, para $n = 50$ puntos (izquierda) y $n = 100$ (derecha).

8.4. Un ejemplo completo

Con una cámara digital doméstica modelo Nikon Coolpix® 3700 se tomaron 23 fotografías del Patio de los Reyes en el interior del Monasterio de San Lorenzo de el Escorial. Todas las imágenes fueron adquiridas sin utilizar el zoom de la cámara, así que se supone que los parámetros intrínsecos son constantes, salvo variaciones debidas al auto-enfoque. Se desconoce la matriz de parámetros intrínsecos. La figura 8.3 muestra dos imágenes de la secuencia.

Se conoce un poco más de la cámara: la distancia focal equivalente empleada durante la adquisición de las imágenes fue de 35 mm (la cámara permite variar la distancia focal entre 35 y 105 mm). Sin embargo, no utilizaremos este conocimiento a durante la calibración.

El objetivo es obtener una reconstrucción euclídea de la escena y una calibración euclídea de las cámaras minimizando el error de reproyección, según se presentó en § 2.3. Seguiremos el esquema de procesamiento de § 2.4. Los pasos son los siguientes:

1. Obtener la calibración proyectiva de las dos cámaras extremas (la N° 1 y la N° 23) y los puntos 3D mediante el algoritmo Gold Standard para la matriz fundamental (§ 6.2.3.6).

2. Estimar las matrices de proyección de las cámaras intermedias (números 2 a 22) mediante el algoritmo Gold Standard de la técnica que hemos llamado resección (§ 5.4.2).

3. Optimizar la calibración proyectiva existente mediante el ajuste de haces proyectivo, minimizando el error de reproyección (§ 6.6.4).

4. Utilizar un algoritmo de autocalibración para calcular el plano del infinito y la matriz de parámetros intrínsecos. En el ejemplo se ha utilizado el algoritmo de estimación de la cuádrica absoluta dual explicado en § 7.6.3.

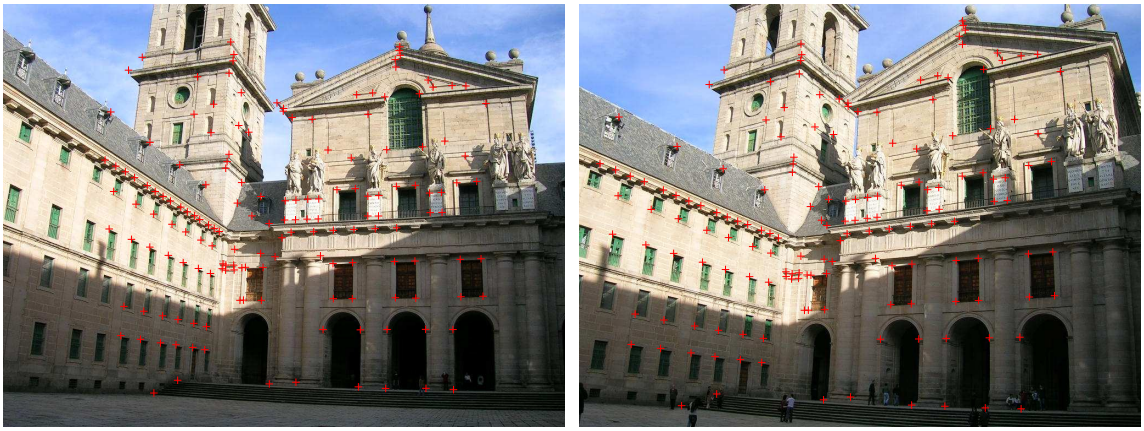


Figura 8.3: Ejemplo con datos reales: imágenes 12 (izquierda) y 21 (derecha) de la secuencia del Patio de los Reyes situado en el interior del Monasterio de San Lorenzo de el Escorial, con las correspondencias de puntos superpuestas (cruces).

5. Actualizar la reconstrucción: construir la homografía del espacio que rectifica los puntos 3D y las matrices de proyección, es decir, los convierte en datos métricos (§ 7.2.3).

6. Optimizar la calibración euclídea actual mediante el ajuste de haces euclídeo, imponiendo la restricción de píxeles cuadrados, minimizando el error de reproyección (§ 8.2).

Tras el ajuste de haces euclídeo faltaría representar la reconstrucción, pero eso no forma parte del proyecto. Tampoco forma parte del proyecto la extracción de características, que se realizó semiautomáticamente, con una herramienta que detecta esquinas en un entorno que determina el usuario.

El error de reproyección normalizado por el número de medidas se calcula según la expresión habitual:

$$\epsilon_{\text{res}}^2 = \frac{\|\widehat{\mathbf{X}} - \mathbf{X}\|^2}{N} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} d(\hat{\mathrm{P}}^i \hat{\mathbf{X}}_j, \mathbf{x}_j^i)^2}{2mn}.$$

Indica el error medio por cada coordenada de los puntos observados. Los errores RMS de reproyección tras cada etapa significativa del método son los mostrados en la tabla 8.1. Los números de los ajustes de haces señalan que la desviación típica presente en los puntos observados es inferior a 1 píxel. Una vez más, se observa lo que predice la teoría: los errores residuales de los ajustes de haces proyectivo y euclídeo son casi iguales, siendo mayor el error en la reconstrucción euclídea. Aunque se ha impuesto la condición de píxeles cuadrados en la última etapa, el error sigue siendo muy pequeño, lo que justifica la hipótesis de píxeles cuadrados.

Etapa	Error reproyección RMS (píxeles)
Calibración proyectiva inicial	0,7399
Tras el ajuste de haces proyectivo	0,5508
Calibración euclídea inicial (píxeles cuadrados)	1,5878
Tras el ajuste de haces euclídeo	0,5528

Cuadro 8.1: Ejemplo con datos reales: errores de reproyección en cada etapa significativa del esquema de procesamiento.

Sólo hacen falta 7 iteraciones del algoritmo de Levenberg-Marquardt en cada uno de los ajuste de haces proyectivo y euclídeo para converger a la solución deseada, lo que indica que el punto de partida

es muy bueno: cae dentro del pozo de atracción del mínimo.

Considérense las dimensiones del problema: $m = 23$ cámaras y $n = 162$ puntos implican que la optimización de la calibración proyectiva (ajuste de haces) se hace con una función modelo

$$f : \mathbb{R}^{762} \longrightarrow \mathbb{R}^{7452},$$

es decir, se busca el mínimo en $\mathbb{R}^{762}$ y ¡sólo se hace en 7 iteraciones del algoritmo LM! Por su parte, el ajuste de haces euclídeo no se queda atrás, el espacio de medidas es el mismo, pero el espacio de parámetros es de dimensión un poco menor

$$f : \mathbb{R}^{693} \longrightarrow \mathbb{R}^{7452}.$$

La salida del ajuste de haces euclídeo son las matrices de proyección y los puntos 3D métricos que minimizan el error de reproyección. De las matrices de proyección podemos extraer las matrices de parámetros intrínsecos de cada imagen (en ellas se cumple que los píxeles son cuadrados, como impone el ajuste de haces).

Las matrices de parámetros intrínsecos de las cámaras N° 12 y N° 21 son:

$$\mathtt{K}^{12} = \begin{bmatrix} 1157{,}9 & 0 & 21{,}4 \\ 0 & 1157{,}9 & -13{,}3 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathtt{K}^{21} = \begin{bmatrix} 1173{,}7 & 0 & -21{,}6 \\ 0 & 1173{,}7 & -29{,}9 \\ 0 & 0 & 1 \end{bmatrix}.$$

En ambas matrices, el punto principal está expresado respecto del centro de la imagen, es decir, el píxel de coordenadas $[512, 384]$ respecto del sistema de referencia habitual de las imágenes (esquina superior izquierda).

Además, se ha probado el ajuste de haces proyectivo con distorsión radial y el error de reproyección obtenido es: $\boldsymbol{\epsilon}_{\text{res}} = 0{,}5367$ píxeles, menor que el error de reproyección sin distorsión radial, como era de esperar, pero la mejora no es espectacular. Los parámetros de distorsión de las cámaras 12 y 21 están recogidos en la tabla 8.2. Para cada cámara, se han utilizado cuatro coeficientes en el polinomio de distorsión radial y las dos coordenadas del centro de distorsión.

Como se aprecia en el error de reproyección y en los parámetros de la cámara, la distorsión radial casi no tiene importancia. Por ejemplo, para la cámara N° 12 hay que situar muy lejos del centro de la imagen el centro de distorsión para disminuir el error de reproyección. En cambio, para la imagen N° 21, el centro de distorsión radial no está tan alejado del centro de la imagen y los coeficientes son mayores que los de la otra cámara, aunque siguen siendo pequeños.

No se debe olvidar que se está evaluando en el punto más propenso a la distorsión, ya que si las imágenes hubiesen sido adquiridas con cualquier otra distancia focal de las que admite la cámara, la distorsión observada sería menor.

Parámetro	Cámara N° 12	Cámara N° 21
x_c	$-706{,}17$	$-145{,}90$
y_c	$110{,}60$	$-130{,}95$
κ_1	$8{,}246 \cdot 10^{-6}$	$2{,}934 \cdot 10^{-6}$
κ_2	$1{,}167 \cdot 10^{-8}$	$-5{,}700 \cdot 10^{-8}$
κ_3	$-1{,}564 \cdot 10^{-11}$	$-1{,}043 \cdot 10^{-10}$
κ_4	$4{,}745 \cdot 10^{-15}$	$1{,}936 \cdot 10^{-13}$

Cuadro 8.2: Ejemplo con datos reales: parámetros de distorsión radial de las dos cámaras consideradas.

También se pueden comparar las diferencias del error de reproyección en las distintas etapas del esquema de procesamiento y la situación excepcional de incluir la distorsión radial mediante los histogramas

del error de reproyección residual. El error de la calibración proyectiva inicial es muy diferente de los otros tres (los tres tipos de ajustes de haces implementados), que son prácticamente iguales. El ruido real no es exactamente gaussiano, como se ha considerado a lo largo de todo el proyecto, pero se parece bastante.

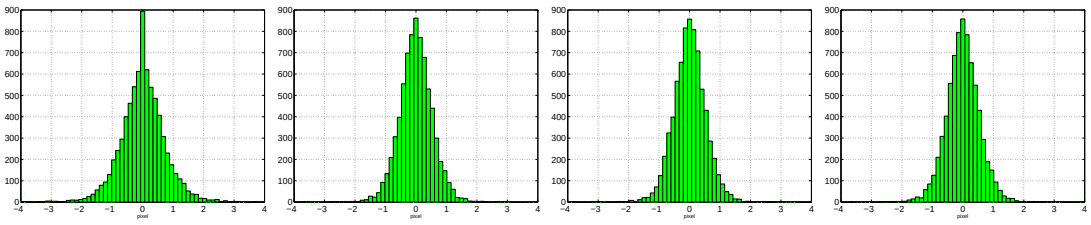


Figura 8.4: Ejemplo con datos reales: histogramas de los errores de reproyección residual. De izquierda a derecha: calibración proyectiva inicial, después del ajustes de haces proyectivo, tras el ajuste de haces proyectivo con distorsión radial (4 coeficientes) y después del ajuste de haces euclídeo. Las respectivas desviaciones típicas del error son: 0,7375, 0,5509, 0,5368 y 0,5529.

Capítulo 9

Resumen y trabajo futuro

Resumen

A lo largo de esta densa memoria se han descrito e implementado numerosos algoritmos de calibración proyectiva, clasificados es tres tipos: lineales, no lineales y robustos. Proporcionan diversidad de combinaciones para obtener la calibración proyectiva de las cámaras a partir de las coordenadas de los puntos y/o rectas en las imágenes.

Dicha calibración proyectiva se puede optimizar mediante un ajuste de haces: una técnica muy beneficiosa para etapas posteriores (autocalibración, etc.). Desarrollando este tema se han descrito algoritmos de ajuste de haces proyectivo y ajuste de haces proyectivo con distorsión radial. Un poco más complicado es extender el ajuste de haces a un marco euclídeo, debido a la parametrización de las matrices de proyección, sin embargo también se ha desarrollado esta idea, dando como resultado un ajuste de haces euclídeo con una parametrización concreta y píxeles cuadrados, aunque también se han dado las bases para utilizar otras parametrizaciones de los elementos del problema.

Los algoritmos lineales se basan en la resolución de un sistema de ecuaciones por mínimos cuadrados, apoyándose en el álgebra matricial. Dan buenos resultados si se normalizan los datos del problema, para evitar inestabilidades numéricas. Se utilizan como inicialización de algoritmos no lineales, los cuales persiguen la minimización de una función de coste mediante esquemas iterativos, basados en la hipótesis de que dicha función es localmente lineal. La ventaja que tienen frente a los lineales es la posibilidad de expresar costes geométricos que guardan relación con medidas realizables en las imágenes.

Los últimos algoritmos en la jerarquía son los robustos, que tratan de mejorar las estimaciones de los algoritmos no lineales sencillos. La alternativa elegida ha sido el esquema RANSAC (desechando otros como mínimos cuadrados ponderados, M-estimadores, Least Median of Squares ...), ya que es sencilla de entender e implementar, además de proporcionar resultados excelentes de acuerdo con los parámetros de diseño del estimador (umbral de clasificación, proporción de outliers esperada, desviación típica de ruido).

Las simulaciones realizadas con datos sintéticos prueban fielmente la relación de los algoritmos descritos con los resultados teóricos esperados y enseñan cuáles son los mejores caminos para alcanzar el objetivo final de la reconstrucción, desechando los métodos menos afortunados.

Trabajo futuro

Sólo resta aplicar los algoritmos a datos reales, igual que se ha hecho en el ejemplo visto en § 8.4, pero en lugar de seleccionar las características semi-automáticamente, se debe poder realizar de forma automática, por lo que es más fácil utilizar secuencias de vídeo para extraer las características, ya que

el desplazamiento de los puntos de una imagen a la siguiente es pequeño.

De la experiencia adquirida tras realizar pruebas con dichas secuencias de vídeo, se concluye que el algoritmo de extracción de características y seguimiento es muy importante, pues el problema suele radicar en las incorrectas correspondencias de puntos (*outliers*), no el ruido de las medidas. En la actualidad se dispone de un algoritmo de extracción de características y seguimiento de puntos ajeno al proyecto, el cual debería ser revisado para mejorar su rendimiento, aunque no es una labor tan grata como la calibración.

Los puntos fuertes del proyecto son, sobre todo, los algoritmos de ajustes de haces, además de los algoritmos de calibración proyectiva y autocalibración implementados. Dichos algoritmos de optimización mejoran mucho el rendimiento de otras etapas y constituyen un paso esencial hacia la consecución de buenos resultados en la calibración y en la reconstrucción final, respectivamente. También podemos decir que el problema de la calibración proyectiva está resuelto, no en vano se ha explicado ese tema en uno de los capítulos más extensos. La distorsión radial también es otro tema que también queda sentenciado, sobre todo porque se ha comprobado que las cámaras domésticas apenas tienen distorsión radial (ejemplo § 8.4), así que sólo debe tenerse en cuenta en situaciones excepcionales. El modelo de proyección lineal es una muy buena aproximación a la realidad.

A pesar de que el proyecto es bastante amplio, todavía quedan muchos asuntos por mejorar. Por ejemplo, habría que crear un interfaz gráfico y unirlo con los dos bloques extremos del esquema de procesamiento dado en § 2.4. Todavía queda un largo camino hasta poder construir un sistema completo de reconstrucción a partir de secuencias de vídeo. El bloque de presentación de la reconstrucción es muy importante porque ofrece resultados visuales de toda la cadena de procesamiento y es el que cuanto antes se debería implementar, ya que en la actualidad se carece del mismo.

Un algoritmo de autocalibración puede ser bueno, pero a ojos de una persona no entendida en el tema, el algoritmo es mucho mejor si le ofreces un modelo tridimensional de la escena reconstruida, con texturas extraídas de las imágenes. Desgraciadamente no he podido presentar tales resultados vistosos, aunque espero conseguirlo en un futuro no muy lejano.

Una vez que se dispone de un esquema completo de procesamiento se puede dar una segunda pasada al esquema y refinar más algunas partes. Por ejemplo, habría que evaluar si es posible atajar el problema de la calibración euclídea desde el principio: en lugar de establecer los bloques separados, tratar de realizar en un esquema conjunto la calibración proyectiva y la autocalibración euclídea.

Deseo final.

Durante el proyecto, a medida que se iban implementando más y más algoritmos surgió la idea de dar forma y sentido al conjunto, completarlo para así crear una librería de rutinas de calibración que sirviera como base sólida para futuros proyectos en el campo de la visión por ordenador, especialmente dentro del Grupo de Tratamiento de Imágenes (GTI). Me gustaría que el trabajo realizado sea útil a otras personas más que el propio autor.

Apéndice A

Notación

Para facilitar la lectura y entendimiento del presente documento se incluyen a continuación algunos criterios de notación seguidos.

Generalidades: habitualmente los vectores están representados mediante letras en negrita, ej. $\mathbf{a}$, y las matrices en letras mayúsculas de fuente teletipo, ej. $\mathtt{A}$.

Cuerpos, espacios y anillos:

$\mathbb{R}$	Cuerpo de los números reales
$\mathbb{C}$	Cuerpo de los números complejos
$\mathbb{P}$	Espacio proyectivo
$\mathbb{S}$	Esfera
$\mathcal{M}_{p\times q}$	Espacio vectorial de las matrices de orden $p \times q$
$\mathcal{M}_{p\times p}$	Anillo de las matrices de orden $p \times p$

Variedades lineales del plano y del espacio proyectivo

$\mathbf{X}$	Punto del espacio
$\mathbf{X}_{\mathrm{M}}$	Punto del espacio en una referencia métrica
π	Plano del espacio
r, $\boldsymbol{\ell}$	Recta del espacio
$\mathbf{x}$	Punto del plano
$\mathbf{x}_j^i$	El j-esimo punto del espacio proyectado en la i-ésima cámara

Generalidades sobre vectores y matrices:

$\mathtt{I}$	Matriz identidad
$\mathtt{H}$	Matriz de una homografía
$[\mathbf{u}]_\times$	Matriz antisimétrica asociada al vector $\mathbf{u}$
C	Matriz de una cónica (en un plano)
$\mathtt{Q}^*$	Matriz de una cuádrica dual
$\|\cdot\|$	Norma 2 de un vector o inducida de una matriz
$\|\cdot\|_F$	Norma Frobenius de una matriz
$\mathcal{U}_F(\mathtt{A}) = \mathtt{A}/\|\mathtt{A}\|_F$	Unitarizar una matriz en norma Frobenius
$\mathtt{A}^\top$	Transpuesta de $\mathtt{A}$
$\mathtt{A}^\perp$	Complemento ortogonal de $\mathtt{A}$
$\mathtt{A}^{-1}$	Inversa de $\mathtt{A}$
$\mathtt{A}^+$	Pseudoinversa de $\mathtt{A}$
$\mathtt{A} \otimes \mathtt{B}$	Producto tensorial de matrices

Otras generalidades.

n, npuntos	Número de puntos
m, ncam	Número de cámaras
Re	Parte real
Im	Parte imaginaria
$\sim$	Igualdad salvo proporcionalidad
λ, k	Constante de proporcionalidad proyectiva
$d(\mathbf{x}_1, \mathbf{x}_2)$	Distancia euclídea entre los puntos de coordenadas homogéneas $\mathbf{x}_1$ y $\mathbf{x}_2$
∇	Primera derivada: gradiente o matriz jacobiana
∇^2	Segunda derivada o matriz hessiana (Hessiano)
F	Función de Lagrange o Lagrangiano

Contexto de las matrices de proyección

$\mathtt{P}$	Matriz de proyección
$\mathtt{P}_\mathrm{M}$	Matriz de proyección métrica
$\mathtt{K}$	Matriz de parámetros intrínsecos
$\mathtt{R}$	Matriz de rotación
$\widetilde{\mathbf{C}}$	Coordenadas afines del centro óptico
κ	Parámetros de distorsión radial
τ	Relación de aspecto
θ	Skew o ángulo de una rotación

Contexto de los algoritmos lineales, la descomposición en valores singulares (SVD), autovalores y autovectores.

$\mathtt{A}$	Matriz de diseño
$\hat{\mathtt{A}}$	Matriz de medida reducida
σ_i	Valor singular i-ésimo
$\mathtt{U}$	Matriz izquierda de la SVD
$\mathtt{D}$	Matriz diagonal de la SVD, con los valores singulares
$\mathtt{V}$	Matriz derecha de la SVD
λ	Autovalor
$\mathbf{\Lambda}$	Matriz de autovalores
$\mathtt{V}$	Matriz de autovectores
$\mathtt{E}$	Matriz de restricciones

Contexto de las geometrías epipolar, trifocal y cuadrifocal.

$\mathbf{e}$	Epipolo
$\mathtt{F}$	Matriz fundamental
$\mathtt{F}_\mathtt{S}$	Parte simétrica de la matriz fundamental
$\mathtt{E}$	Matriz esencial
$\mathcal{T}$	Tensor trifocal
$\mathtt{T}_i$	Tensor trifocal en forma de matrices 3×3
$\mathbf{t}$	Tensor trifocal en forma de vector
ϵ_{ijk}	Tensor de permutación
Q	Tensor cuadrifocal
$\hat{Q}$	Tensor cuadrifocal reducido
$\mathbf{q}$	Tensor cuadrifocal en forma de vector

Contexto de los elementos de autocalibración

π_∞	Plano del infinito
Ω_∞	Cónica absoluta en el plano del infinito
$\mathtt{Q}^*_\infty$	Cuádrica absoluta dual
$\mathtt{H}_\infty$	Homografía de infinito
ω	IAC o PAC, proyección de la cónica absoluta
ω^*	Dual de la proyección de la cónica absoluta
Ω	Cuádrica de Klein
Σ	Cuádrica degenerada del Calibration Pencil
$\mathbf{h}(\theta)$	Horóptera parametrizada
θ	Parámetro de una horóptera

Contexto de optimización Levenberg-Marquardt

f	Función modelo
M	Dimensión del espacio de parámetros: número de variables "independientes"
N	Dimensión del espacio de medidas: número de variables "dependientes"
$\mathbf{P}$	Vector de parámetros
$\mathbf{X}$	Vector de medidas objetivo
$\Sigma_{\mathbf{X}}$	Matriz de covarianzas del vector de medidas
$\widehat{\mathbf{X}}$	Vector de medidas estimado
ϵ	Vector de error o residuo
$\mathtt{J}$	Matriz jacobiana
$\mathtt{A}$	Parte izquierda de una matriz jacobiana
$\mathtt{B}$	Parte derecha de una matriz jacobiana
$\mathtt{U}$	Parte de la matriz $\mathtt{J}^\top\mathtt{J}$ dependiente de las cámaras
$\mathtt{V}$	Parte de la matriz $\mathtt{J}^\top\mathtt{J}$ dependiente de los puntos 3D
$\mathtt{W}$	Parte de la matriz $\mathtt{J}^\top\mathtt{J}$ dependiente de cámaras y puntos 3D
$\mathtt{H}$	Matriz hessiana (Hessiano)

Contexto de ruido

σ	Desviación típica
$\boldsymbol{\epsilon}_{\text{res}}$	Error residual (distancia de los datos estimados a los datos ruidosos)
$\boldsymbol{\epsilon}_{\text{est}}$	Error de estimación (distancia de los datos estimados a los datos exactos)
d	Grados de libertad, número de parámetros en la fórmula de Cramer-Rao

Apéndice B

Descomposición en Valores Singulares (SVD)

El objetivo de este capítulo es recopilar la información básica necesaria para entender la Descomposición en Valores Singulares (SVD). Empezamos dando la definición de la SVD para una matriz rectangular genérica A, y discutiendo algunos conceptos relacionados [28]. Después ilustramos tres importantes aplicaciones de la SVD:

- Resolución de sistemas de ecuaciones lineales no homogéneos.
- Resolución de sistemas de ecuaciones lineales homogéneos de rango deficiente.
- Garantizar que los elementos de una matriz estimada numéricamente satisfacen alguna restricción dada (por ejemplo, ortogonalidad).

B.1. Definición

Singular Value Decomposition (SVD)

Cualquier matriz rectangular de $m \times n$ se puede escribir como el producto de tres matrices:

$$A = UDV^{\top}. \tag{B.1}$$

Las columnas de la matriz cuadrada U de $m \times m$ son vectores unitarios mutuamente ortogonales, así como las columnas de la matriz V de $n \times n$. (Las matrices U y V son ortogonales: $U^{\top}U = I_m$ y $V^{\top}V = I_n$). La matriz D de $m \times n$ es diagonal. Los elementos de su diagonal, σ_i, son los llamados valores singulares, y son tales que $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_n \geq 0$.
Para el caso de sistemas sobredeterminados con los que solemos trabajar $m > n$, queda:

$$D = \begin{bmatrix} \sigma_1 & & & & 0 \\ & \sigma_2 & & & \\ & & \ddots & & \\ & & & \sigma_{n-1} & \\ 0 & & & & \sigma_n \\ & & 0 & & \\ & & \vdots & & \\ & & 0 & & \end{bmatrix}$$

Aunque U y V no son únicas, los valores singulares σ_i están completamente determinados por A.

B.2. Propiedades de la SVD

Propiedad 1
Los valores singulares dan información valiosa acerca de la singularidad de una matriz cuadrada. Una matriz cuadrada, A, es regular (o no singular) si y sólo si todos sus valores singulares son no nulos. Más aún, los σ_i también dan información de cuan próxima está una matriz de ser singular: la razón

$$C = \frac{\sigma_1}{\sigma_n},$$

llamada número de condición, mide el grado de singularidad de A. Cuando $1/C$ es comparable con la precisión aritmética del ordenador, la matriz A está débilmente condicionada, y para todos los propósitos prácticos puede ser considerada singular.

Propiedad 2
Si A es un matriz rectangular, el número de σ_i no nulos es igual al rango de la matriz A. Así, dada una tolerancia fija, ϵ, el número de valores singulares mayores que ϵ es igual al rango efectivo de la matriz.

Propiedad 3
Si A es una matriz cuadrada no singular, su inversa se puede escribir como:

$$A^{-1} = VD^{-1}U^\top$$

Sea A singular o no, la pseudoinversa de A, A^+, se puede escribir como

$$A^+ = VD_0^{-1}U^\top, \tag{B.2}$$

con D_0^{-1} igual a D^{-1} para todo valor singular no nulo y cero en caso contrario. Si A es regular, entonces $D_0^{-1} = D^{-1}$ y $A^+ = A^{-1}$.

Propiedad 4
Las columnas de U correspondientes a los valores singulares no nulos forman una base ortonormal del recorrido de A. Las columnas de V correspondientes a los valores singulares nulos forman una base ortonormal para el núcleo de A.

Propiedad 5
Los cuadrados de los valores singulares no nulos son los autovalores no nulos de las matrices $A^\top A$ de $n \times n$ y $AA^\top$ de $m \times m$. Las columnas de U son los autovectores de $AA^\top$, las columnas de V son los autovectores de $A^\top A$. Además,

$$\begin{aligned} A^\top \mathbf{u}_k &= \sigma_k \mathbf{v}_k \\ A\mathbf{v}_k &= \sigma_k \mathbf{u}_k, \end{aligned}$$

donde $\mathbf{u}_k$ y $\mathbf{v}_k$ son las columnas de U y V asociadas a σ_k.

Para ver la conexión de los valores singulares de A con autovalores de $A^\top A$, hacemos el siguiente razonamiento: si $A = UDV^\top$, entonces $A^\top A = VDU^\top UDV^\top = VD^2V^\top$ porque U es ortogonal y D diagonal. Como V también lo es, $V^\top = V^{-1}$, así que $A^\top A = VD^2V^{-1}$. Esta es la ecuación de definición de los autovalores, que indica que los elementos de D^2 son los autovalores de $A^\top A$ y las columnas de V son los autovectores de $A^\top A$. En resumen: los valores singulares son la raíz cuadrada de los autovalores de $A^\top A$.

No hay que olvidar que $A^\top A$ es simétrica y semidefinida positiva, así que sus autovalores son reales y no negativos. Consecuentemente, los valores singulares son reales y no negativos.

Propiedad 6
Una posible medida de distancia entre matrices puede obtenerse a partir de la norma Frobenius. La norma Frobenius de una matriz A es, simplemente, la raíz cuadrada de la suma de los cuadrados de los elementos a_{ij} de A:

$$\|A\|_F = \sqrt{\sum_i \sum_j a_{ij}^2}. \tag{B.3}$$

Sustituyendo la Descomposición en Valores Singulares (B.1) en (B.3) y dado que el producto por matrices unitarias preserva la norma euclídea de cada fila o columna, y por tanto preserva $\|\cdot\|_F$, se llega a:

$$\|A\|_F = \sqrt{\sum_i \sigma_i^2}.$$

Ejemplo
Consideremos la matriz

$$A = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 2 & 2 \end{bmatrix}$$

Calculamos

$$A^\top A = \begin{bmatrix} 9 & 9 \\ 9 & 9 \end{bmatrix},$$

cuyos autovalores son:

$$\det(A^\top A - \lambda I) = (9-\lambda)^2 - 9^2 = \lambda^2 - 18\lambda = 0$$
$$\lambda_1 = 18,\ \lambda_2 = 0$$

Comprobamos que ambos autovalores son no negativos porque $A^\top A$ es semidefinida positiva. Por la propiedad 5, los valores singulares son:

$$\sigma_1 = \sqrt{18},\ \sigma_2 = 0$$

Los dos autovectores normalizados (de norma unidad) de $A^\top A$ son:

$$\mathbf{v}_1 = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}^\top$$
$$\mathbf{v}_2 = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix}^\top$$

Con estos vectores podemos construir la matriz $V = [\mathbf{v}_1\ \mathbf{v}_2]$. A continuación, procedemos a calcular la matriz U, mediante el cálculo de los autovectores normalizados de $AA^\top$.

$$AA^\top = \begin{bmatrix} 2 & 4 & 4 \\ 4 & 8 & 8 \\ 4 & 8 & 8 \end{bmatrix}$$

Los autovalores son:

$$\det(AA^\top - \lambda I) = \lambda^3 - 18\lambda^2 = 0$$
$$\lambda_1 = 18, \lambda_2 = 0, \lambda_3 = 0.$$

Vemos que $\lambda_1 = \sigma_1^2$ y $\lambda_2 = \sigma_2^2$. Los autovectores normalizados son:

$$\mathbf{u}_1 = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \end{bmatrix}^\top$$
$$\mathbf{u}_2 = \begin{bmatrix} -\frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \end{bmatrix}^\top$$
$$\mathbf{u}_3 = \begin{bmatrix} -\frac{2\sqrt{5}}{5} & 0 & \frac{\sqrt{5}}{5} \end{bmatrix}^\top$$

con los cuales se forma $U = [\mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3]$. Ahora podemos comprobar que $A = UDV^\top$:

$$UDV^\top = \begin{bmatrix} \frac{1}{3} & -\frac{2\sqrt{5}}{5} & -\frac{2\sqrt{5}}{5} \\ \frac{2}{3} & \frac{\sqrt{5}}{5} & 0 \\ \frac{2}{3} & 0 & \frac{\sqrt{5}}{5} \end{bmatrix} \begin{bmatrix} 3\sqrt{2} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 2 & 2 \end{bmatrix} = A$$

También comprobamos que $A\mathbf{v}_i = \sigma_i \mathbf{u}_i$ y que $A^\top \mathbf{u}_i = \sigma_i \mathbf{v}_i$, siendo $i = 1, 2$ (propiedad 5). Para $i = 2$, ambos resultados son el vector cero porque $\sigma_2 = 0$, en cambio, para $i = 1$, obtenemos:

$$A\mathbf{v}_1 = \left[\sqrt{2}\ \ 2\sqrt{2}\ \ 2\sqrt{2}\right]^\top = \sqrt{18}\left[\frac{1}{3}\ \ \frac{2}{3}\ \ \frac{2}{3}\right]^\top = \sigma_1 \mathbf{u}_1$$

y

$$A^\top \mathbf{u}_1 = [3\ \ 3]^\top = \sqrt{18}\left[\frac{\sqrt{2}}{2}\ \ \frac{\sqrt{2}}{2}\right]^\top = \sigma_1 \mathbf{v}_1.$$

B.3. Aplicaciones de la SVD

B.3.1. Mínimos Cuadrados

Supongamos que tenemos que resolver un sistema de m ecuaciones lineales,

$$A\mathbf{x} = \mathbf{b}, \tag{B.4}$$

donde la incógnita es el vector n-dimensional $\mathbf{x}$. La matriz A de tamaño $m \times n$ contiene los coeficientes del sistema y el vector m-dimensional $\mathbf{b}$ son los datos. Si no todos los elementos del vector $\mathbf{b}$ son nulos, la solución se puede obtener multiplicando ambos lados de la ecuación por $A^\top$, llegando a las Ecuaciones Normales:

$$A^\top A\mathbf{x} = A^\top \mathbf{b}. \tag{B.5}$$

La solución viene dada por

$$\mathbf{x} = (A^\top A)^+ A^\top \mathbf{b}.$$

Esta solución es óptima en el sentido de mínimos cuadrados. Minimiza la norma del vector $\|A\mathbf{x} - \mathbf{b}\|$, llamado residuo; por eso se la llama solución mínimo-cuadrática del sistema.

Normalmente es una buena idea calcular las pseudoinversa de $A^\top A$ a través de la SVD. En el caso de más ecuaciones que incógnitas la pseudoinversa tiende a coincidir con la inversa de $A^\top A$, pero no hay que perder de vista el número de condición de $A^\top A$ (Propiedad 1).

Casos particulares y algoritmos

Sistemas de rango completo
Si la matriz A, de tamaño $m \times n$, cumple $rango(A) = n$ y no existe una solución exacta al sistema, buscamos un vector solución $\mathbf{x}$ tal que minimice $\|A\mathbf{x} - \mathbf{b}\|$.

Buscamos un $\mathbf{x}$ que minimice $\|A\mathbf{x} - \mathbf{b}\| = \|UDV^\top \mathbf{x} - \mathbf{b}\|$. Por la propiedad de preservar la norma de las matrices ortogonales, $\|UDV^\top \mathbf{x} - \mathbf{b}\| = \|DV^\top \mathbf{x} - U^\top \mathbf{b}\|$, y ésta es la cantidad que queremos minimizar. Escribiendo $\mathbf{y} = V^\top \mathbf{x}$ y $\mathbf{b}' = U^\top \mathbf{b}$, el problema se convierte en minimizar $\|D\mathbf{y} - \mathbf{b}'\|$. Este

sistema es de la forma

$$\begin{bmatrix} \sigma_1 & & & & 0 \\ & \sigma_2 & & & \\ & & \ddots & & \\ & & & \sigma_{n-1} & \\ 0 & & & & \sigma_n \\ & & 0 & & \\ & & \vdots & & \\ & & 0 & & \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_{n-1} \\ b'_n \\ b'_{n+1} \\ \vdots \\ b'_m \end{bmatrix}.$$

Vemos claramente que el vector $D\mathbf{y}$ más cercano a $\mathbf{b}'$ es el vector $[b'_1, b'_2, \ldots, b'_n, 0, \ldots, 0]^\top$. Esto se consigue haciendo $y_i = b'_i/\sigma_i$, para $i = 1, \ldots, n$. Recordemos que, al haber supuesto que $rango(A) = n$, esto asegura que ningún valor singular es nulo ($\sigma_i \neq 0$). Finalmente se deshace el cambio $\mathbf{y} = V^\top \mathbf{x}$, para despejar: $\mathbf{x} = V\mathbf{y}$. El algoritmo completo es:

OBJETIVO

Encontrar la solución mínimo-cuadrática del sistema $A\mathbf{x} = \mathbf{b}$, donde A es de tamaño $m \times n$, con $m > n$ (sobredeterminado), y $rango(A) = n$.

ALGORITMO

1. Hallar la SVD $A = UDV^\top$
2. Hacer el cambio $\mathbf{b}' = U^\top \mathbf{b}$
3. Hallar el vector $\mathbf{y}$ definido por $y_i = b'_i/\sigma_i$, siendo σ_i el i-ésimo valor singular de A.
4. La solución es $\mathbf{x} = V\mathbf{y}$.

Sistemas de rango deficiente

Sea A de tamaño $m \times n$ y se cumple $r = rango(A) < n$. Es posible que por efecto del ruido la matriz A tenga rango mayor que n, pero queremos forzar la restricción de rango r por consideraciones teóricas del problema particular que estemos considerando. En este caso habrá una familia de soluciones al sistema de ecuaciones, la cual depende de $(n - r)$ parámetros. Podemos obtener fácilmente la familia mediante la SVD:

OBJETIVO

Encontrar la solución general del sistema $A\mathbf{x} = \mathbf{b}$, donde A es de tamaño $m \times n$ ($m > n$) (sobredeterminado) y $rango(A) = r < n$. Solución en el sentido de minimizar $\|A\mathbf{x} - \mathbf{b}\|$.

ALGORITMO

1. Hallar la SVD $A = UDV^\top$ (con los valores singulares en el orden habitual: descendente).
2. Hacer el cambio $\mathbf{b}' = U^\top \mathbf{b}$.
3. Hallar el vector $\mathbf{y}$, de dimensión n, definido por

$$y_i = \begin{cases} b'_i/\sigma_i & i = 1, \ldots, r \\ 0 & \text{resto} \end{cases}$$

   Siendo σ_i el i-ésimo valor singular de A.
4. La solución particular $\mathbf{x}_0$ de mínima norma $\|\mathbf{x}\|$ es $\mathbf{x}_0 = V\mathbf{y}$.
5. La solución general es $\mathbf{x} = V\mathbf{y} + \lambda_{r+1}\mathbf{v}_{r+1} + \cdots + \lambda_n \mathbf{v}_n$, donde $\mathbf{v}_{r+1}, \ldots, \mathbf{v}_n$ son las últimas $n - r$ columnas de V.

Se puede escribir más compactamente definiendo una matriz que controle el paso 3. Sea D la matriz diagonal de $m \times n$ que resulta de la SVD. Por ser diagonal y de rango deficiente cumple: $D_{ij} = 0$ para $i \neq j$ y $D_{ii} = \sigma_i$ para todo i; en donde $\sigma_i \neq 0$ para $1 \leq i \leq r$ y $\sigma_i = 0$ para $r+1 \leq i \leq n$. Entonces, D_0^{-1} es aquella matriz de $n \times m$ (nótese la inversión de índices) cuyos únicos elementos no nulos son $D_{0\,ii}^{-1} = 1/\sigma_i$ para $1 \leq i \leq r$. Esta matriz es la que vimos en la propiedad 3 de la SVD (ecuación B.2).

Con esta notación, la solución de mínima norma $\|\mathbf{x}\|$ que minimiza $\|A\mathbf{x} - \mathbf{b}\|$ es $\mathbf{x}_0 = VD_0^{-1}U^\top \mathbf{b}$, que como ya vimos es el producto de la matriz pseudoinversa $A^+ = VD_0^{-1}U^\top$ por el vector $\mathbf{b}$.

$$\mathbf{x}_0 = VD_0^{-1}U^\top \mathbf{b} = A^+ \mathbf{b}$$

El algoritmo da una familia de soluciones, parametrizadas por los $(n-r)$ parámetros λ_i.

Ejemplo
Consideremos el problema de mínimos cuadrados $A\mathbf{x} = \mathbf{b}$:

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 15 \\ 15 \\ -30 \end{bmatrix}.$$

De la matriz A ya conocemos la Descomposición en Valores Singulares porque la hemos hecho en el ejemplo anterior. Siguiendo el algoritmo, se calcula $\mathbf{b}' = U^\top \mathbf{b}$, $\mathbf{y} = D_0^{-1}\mathbf{b}'$ y entonces $\mathbf{x} = V\mathbf{y}$.

$$\mathbf{b}' = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ -\frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ -\frac{2\sqrt{5}}{5} & 0 & \frac{\sqrt{5}}{5} \end{bmatrix} \begin{bmatrix} 15 \\ 15 \\ -30 \end{bmatrix} = \begin{bmatrix} -5 \\ -3\sqrt{5} \\ -12\sqrt{5} \end{bmatrix},$$

$$\mathbf{y} = \begin{bmatrix} \frac{1}{3\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -5 \\ -3\sqrt{5} \\ -12\sqrt{5} \end{bmatrix} = \begin{bmatrix} -\frac{5\sqrt{2}}{6} \\ 0 \end{bmatrix},$$

$$\mathbf{x} = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} -\frac{5\sqrt{2}}{6} \\ 0 \end{bmatrix} = \begin{bmatrix} -\frac{5}{6} \\ -\frac{5}{6} \end{bmatrix}.$$

También se puede calcular la pseudoinversa $A^+ = VD_0^{-1}U^\top$:

$$A^+ = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} \frac{1}{3\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ -\frac{2\sqrt{5}}{5} & \frac{\sqrt{5}}{5} & 0 \\ -\frac{2\sqrt{5}}{5} & 0 & \frac{\sqrt{5}}{5} \end{bmatrix} = \begin{bmatrix} \frac{1}{18} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{18} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

Entonces se calcula $\mathbf{x}$ directamente como $\mathbf{x} = A^+\mathbf{b}$:

$$\mathbf{x} = \begin{bmatrix} \frac{1}{18} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{18} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} \begin{bmatrix} 15 \\ 15 \\ -30 \end{bmatrix} = \begin{bmatrix} -\frac{5}{6} \\ -\frac{5}{6} \end{bmatrix},$$

al igual que antes. Todas las demás soluciones son de la forma:

$$\mathbf{x} = \begin{bmatrix} -\frac{5}{6} \\ -\frac{5}{6} \end{bmatrix} - \lambda_2 \begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix}.$$

Sistemas de rango desconocido
Si el rango del sistema de ecuaciones no es conocido (no tiene ninguna restricción), es apropiado poner a cero los valores singulares que sean pequeños comparados con el mayor valor singular. Así, si $\sigma_i/\sigma_1 < \delta$, donde δ es una constante del orden de la precisión del ordenador[1], entonces se pone $y_i = 0$. Se obtiene una solución mínimo-cuadrática al igual que el caso anterior: $\mathbf{x} = V\mathbf{y}$.

B.3.2. Sistemas Homogéneos

Supongamos que queremos resolver un sistema homogéneo de m ecuaciones y n incógnitas

$$A\mathbf{x} = \mathbf{0}, \tag{B.6}$$

con $m > n$ y $rango(A) = n-1$. Descartando la solución trivial $\mathbf{x} = \mathbf{0}$, una solución, a falta de un factor de escala, se puede obtener fácilmente mediante la SVD. Esta solución es proporcional al autovector correspondiente al único autovalor nulo de $A^\top A$ (el resto de autovalores son estrictamente positivos porque $rango(A) = n - 1$).

Demostración
Debido a que la norma de la solución de un sistema de ecuaciones homogéneo es arbitraria, buscamos una solución de norma unidad y en el sentido de mínimos cuadrados. Por lo tanto, queremos minimizar

$$\|A\mathbf{x}\|^2 = (A\mathbf{x})^\top A\mathbf{x} = \mathbf{x}^\top A^\top A\mathbf{x},$$

sujetos a la restricción

$$\|\mathbf{x}\|^2 = \mathbf{x}^\top \mathbf{x} = 1.$$

Introduciendo el multiplicador de Lagrange λ, esto es equivalente a minimizar sin restricciones el llamado Lagrangiano

$$\mathcal{L}(\mathbf{x}) = \mathbf{x}^\top A^\top A\mathbf{x} - \lambda(\mathbf{x}^\top \mathbf{x} - 1).$$

Igualando a cero la derivada del Lagrangiano respecto de $\mathbf{x}$ (el gradiente) resulta

$$A^\top A\mathbf{x} - \lambda\mathbf{x} = \mathbf{0}.$$

Esta ecuación nos indica que λ es un autovalor de $A^\top A$, y la solución, $\mathbf{x} = \mathbf{e}_\lambda$, el autovector asociado. Reemplazando $\mathbf{x}$ por $\mathbf{e}_\lambda$, y $A^\top A\mathbf{e}_\lambda$ por $\lambda\mathbf{e}_\lambda$ el Lagrangiano queda:

$$\mathcal{L}(\mathbf{e}_\lambda) = \mathbf{e}_\lambda^\top \lambda\mathbf{e}_\lambda = \lambda\mathbf{e}_\lambda^\top \mathbf{e}_\lambda = \lambda.$$

Por lo tanto, el mínimo se alcanza en $\lambda = 0$, el menor autovalor de $A^\top A$. Pero de las propiedades 4 y 5, se sigue que esta solución podría haber sido establecida de forma equivalente como la columna de V correspondiente al único valor singular nulo de A (el núcleo de A).

Se suele aplicar a sistemas de rango deficiente en los que A no es singular por errores numéricos o ruido, pero debería serlo (ej: matriz `F`). En este caso la solución es la columna de V correspondiente al menor valor singular de A (última columna de V porque los valores singulares están ordenados).

[1] La precisión de una máquina es el mayor valor en coma flotante ϵ tal que $1{,}0 + \epsilon = 1{,}0$.

B.3.3. Imposición de restricciones

Frecuentemente se generan estimaciones numéricas de una matriz, A, cuyos elementos no son todos independientes, sino que satisfacen alguna restricción algebraica. Este es el caso, por ejemplo, de las matrices ortogonales o la matriz fundamental. Lo que suele ocurrir es que los errores introducidos por ruido y operaciones numéricas alteran la matriz estimada. Esto puede causar serios problemas si los posteriores algoritmos que utilicen la matriz asumen que ésta satisface exactamente las restricciones.

Primer problema de minimización con restricciones
Supongamos que queremos resolver el sistema homogéneo $A\mathbf{x} = \mathbf{0}$, y que podemos describir las restricciones que se deben satisfacer mediante el sistema $C\mathbf{x} = \mathbf{0}$. El problema que se plantea es

> ◇ Encontrar el $\mathbf{x}$ que minimiza $\|A\mathbf{x}\|$ sujeto a las restricciones $\|\mathbf{x}\| = 1$ y $C\mathbf{x} = \mathbf{0}$.

Este problema se puede resolver transformándolo en uno del que ya conocemos su solución: un sistema homogéneo de ecuaciones sin restricciones. La condición de que $\mathbf{x}$ satisface $C\mathbf{x} = \mathbf{0}$ significa que $\mathbf{x}$ es perpendicular a todas las filas de C. El conjunto de todos estos $\mathbf{x}$ es el espacio vectorial llamado complemento ortogonal del espacio generado por las filas de C.

Primero, si C tiene menos filas que columnas, entonces la extendemos hasta convertirla en una matriz cuadrada, simplemente añadiendo filas con ceros. Esto no tiene efecto en el conjunto de restricciones $C\mathbf{x} = \mathbf{0}$.

Sea $C = UDV^\top$ la Descomposición en Valores Singulares de C, donde D es una matriz diagonal con r elementos no nulos. En este caso, C tiene rango r y el espacio fila de C es el generado por las primeras r filas de $V^\top$. El complemento ortogonal del espacio fila de C se genera con las restantes filas de $V^\top$. Definamos $C^\perp$ como la matriz V sin sus primeras r columnas. Entonces $CC^\perp = 0$, y el conjunto de vectores $\mathbf{x}$ que satisfacen $C\mathbf{x} = \mathbf{0}$ es el generado por las columnas de $C^\perp$ y podemos escribir cualquier vector como $\mathbf{x} = C^\perp\mathbf{x}'$, para cierto $\mathbf{x}'$. Como $C^\perp$ tiene sus columnas ortogonales ($C^\perp$ es ortogonal), se observa que $\|\mathbf{x}\| = \|C^\perp\mathbf{x}'\| = \|\mathbf{x}'\|$. Ya podemos formular el problema de minimización que conocemos:

> ◇ Encontrar el $\mathbf{x}'$ que minimiza $\|AC^\perp\mathbf{x}'\|$ sujeto a la restricción $\|\mathbf{x}'\| = 1$ (para no obtener la solución trivial).

Es decir, se han incluido las restricciones dentro de la norma del vector cuyo módulo queremos minimizar. La solución, $\mathbf{x}$, se obtiene en dos pasos:

1. Mediante la SVD de $AC^\perp = UDV^\top$, hallamos $\mathbf{x}'$: el vector columna de V correspondiente al menor valor singular de $AC^\perp$, es decir, la última columna de V.
2. Deshacemos el cambio: $\mathbf{x} = C^\perp\mathbf{x}'$

Segundo problema de minimización con restricciones
Llamemos $\hat{A}$ a la matriz cuyas entradas no satisfacen las restricciones dadas. Una vez más, la SVD resulta ser una herramienta útil; nos permite encontrar la matriz más próxima a $\hat{A}$, en el sentido de la norma inducida (propiedad 6), que satisface las restricciones de forma exacta. Esto se logra calculando la SVD de la matriz estimada, $\hat{A} = UDV^\top$, y estimando A como $UD'V^\top$, con D' obtenida cambiando los valores singulares de D por los esperados cuando se satisfacen las restricciones. [2] Entonces, los elementos de $A = UD'V^\top$ satisfacen las restricciones de forma exacta por propia construcción.

B.4. Complejidad computacional de la SVD

La complejidad computacional de la Descomposición en Valores Singulares depende de cuánta información queramos obtener. Ya hemos indicado que puede que sólo nos interese el vector columna de V asociado al menor valor singular. En este caso, la matriz U no se usa y no necesitamos calcularla. Si,

[2] Si $\hat{A}$ es una buena estimación numérica, sus valores singulares no estarán muy alejados de los esperados.

por el contrario, queremos la descomposición completa, aumenta la carga computacional. Los sistemas de ecuaciones sobredeterminados (la matriz del sistema tiene más filas que columnas) el esfuerzo extra que requiere calcular la matriz U es considerable.

Demos algunas expresiones para el número de operaciones en coma flotante (flops) necesarias para calcular la SVD de una matriz de $m \times n$. Para hallar las matrices U, V y D, se necesitan un total de $4m^2n + 8mn^2 + 9n^3$. Si sólo se necesita conocer las matrices V y D, entonces sólo son necesarias $4mn^2 + 8n^3$ flops. Esta distinción es importante porque la última expresión no contiene términos en m^2 ($m \gg n$). El número de operaciones necesarias para calcular U varía según el cuadrado de m, el número de filas. Por el contrario, la complejidad de calcular D y V es lineal con m. En los casos en que hay muchas más filas que columnas (sistemas sobredeterminados), es importante evitar calcular U, a menos que sea imprescindible.

Para ampliar conocimientos sobre este tema, se puede consultar [1, pág. 557-558].

Apéndice C

Rutinas de Optimización del *Numerical Recipes in C*

C.1. Introducción

En el presente capítulo se comentan las rutinas del capítulo 10 de [31], "Minimization or Maximization of Functions", las cuales han sido traducidas a MATLAB para su posible utilización: para no estar vendidos a lo que dicte el paquete de optimización de MATLAB (Optimization Toolbox).

El planteamiento es el siguiente: nos dan una función que depende de una o más variables independientes $f : \mathbb{R}^M \to \mathbb{R}$. Queremos encontrar el valor de esas variables donde f toma una valor mínimo o máximo, junto con este valor. Se trata, pues, de una optimización sin restricciones (unconstrained optimization): la búsqueda se efectúa sobre todo el espacio de parámetros o variables independientes. Únicamente se considera el caso de minimización, pues la maximización de f es lo mismo que la minimización de $-f$.

El citado capítulo constituye una selección de los algoritmos mejor establecidos para minimización sin restricciones. Algunos algoritmos están relacionados entre sí, por lo que no se pueden considerar rutinas aisladas. Desgraciadamente, no existe el algoritmo perfecto de minimización, por lo que no se debe uno limitar a utilizar una sola rutina, sino que se pueden comparar varias. La elección inicial del algoritmo puede estar basada en las siguientes consideraciones:

- Se debe elegir entre métodos que necesitan sólo evaluaciones de la función a minimizar o métodos que también requieren evaluaciones de la derivada de la función. En el caso multidimensional, dicha derivada es el gradiente. Los métodos que utilizan la derivada son, de alguna forma, más poderosos que los que sólo conocen la función, pero a veces no tanto como para compensar el coste del cálculo de derivadas.
- Para una minimización unidimensional (minimizar una función de una variable) sin cálculo de la derivada, lo recomendable es acotar el intervalo donde se encuentra el mínimo con `mnbrak` y después utilizar el método de Brent programado en `brent`. Si la derivada segunda de la función (o la primera derivada) es discontinua, entonces, las interpolaciones parabólicas del método de Brent no suponen ninguna ventaja, y se puede utilizar la forma más sencilla de la búsqueda por la razón áurea (golden section search).
- Para una minimización unidimensional con cálculo de la derivada, `dbrent` implementa una variante del método de Brent que hace un uso limitado de la información de la derivada. Se desestima la posibilidad de utilizar la derivada para construir polinomios interpoladores de la función de orden superior a 2.

Pasamos al caso multidimensional, con y sin cálculo de la derivada. $f : \mathbb{R}^M \to \mathbb{R}$.

- Se debe elegir entre los métodos que requieran almacenamiento (memoria) de orden M^2 y los que sólo lo necesitan de orden M. Para valores moderados de M y memorias de tamaño razonable esta no es una limitación seria. Habrá, sin embargo, ocasiones en las que el gasto de memoria sea crítico.

- Se implementa el método del símplex cuesta abajo (downhill simplex method) de Nelder y Mead en `amoeba`. En algunos casos puede ser extremadamente lento, pero es muy robusto. Es muy útil cuando el cálculo del mínimo es una cosa accesoria (sin importancia) dentro del problema general. El gasto de memoria requerido es de orden M^2 y no necesita cálculo de la derivada.

- También se incluye, en `powell`, el método de Powell, prototipo de los métodos que utilizan un conjunto de direcciones. Son los métodos a elegir cuando no se pueden calcular fácilmente la derivada y tampoco se deberían despreciar en caso contrario. A pesar de que no se necesita proporcionar una rutina que indique la derivada, el método necesita una subrutina de minimización unidimensional como el método de Brent. El almacenamiento es de orden M^2.

Existen dos grandes familias de algoritmos de minimización multidimensional con cálculo de las primeras derivadas. Ambas familias necesitan un sub-algoritmo de minimización unidimensional, el cual puede a su vez utilizar o no la información de la derivada, según se juzgue conveniente (dependiendo del esfuerzo necesario para calcular la función y su gradiente). Son distintas alternativas: ninguna familia domina sobre la otra.

- La primera familia se conoce bajo el nombre de métodos del gradiente conjugado (conjugate gradient methods), según representa el algoritmo de Fletcher-Reeves y el estrechamente relacionado y probablemente superior algoritmo de Polak-Ribiere. Estos métodos requieren almacenamiento de orden pocas veces M, cálculo de la derivada y sub-minimizaciones unidimensionales.

- La segunda familia se conoce bajo el nombre de métodos de métrica variable o cuasi-Newton, como representa el algoritmo de Davidon-Fletcher-Powell (DFP) o el estrechamente relacionado algoritmo de Broyden-Fletcher-Goldfarb-Shanno (BFGS). Estos métodos requieren memoria de orden M^2, cálculo de la derivada y sub-minimizaciones unidimensionales.

De la segunda familia no se ha programado ninguna rutina, sin embargo, se ha implementado un algoritmo que se aleja un poco del resto: el algoritmo de temple o enfriamiento simulado (simulated annealing). Esta familia es relativamente nueva y ha resuelto algunos problemas que previamente se pensaba eran irresolubles; enfocan el problema de encontrar un mínimo global en presencia de muchos mínimos locales no deseados.

C.2. Métodos unidimensionales

C.2.1. Búsqueda mediante la Razón Áurea

Se considera que en un intervalo hay un mínimo si para tres puntos $a < b < c$ se cumple que $f(b)$ es menor que $f(a)$ y que $f(c)$. En este caso, sabemos que la función (si no es singular) tiene un mínimo en el intervalo (a, c).

El método de búsqueda consiste en iterar con el anterior esquema: a partir de una tupla $[a, b, c]$ se elige un nuevo punto x entre a y b o entre b y c, y se actualiza la tupla a $[a, x, b]$ o $[b, x, c]$ en función de $f(x)$. En cada iteración, se ordena la tupla para que el valor central de la misma contenga la abscisa cuya ordenada es el mejor mínimo hasta el momento. El proceso continua hasta que la distancia entre las abscisas extremas de la tupla alcance una tolerancia.

Sea $\Delta = c - a$ la anchura del intervalo (a, c). Se demuestra que el intervalo óptimo $[a, b, c]$ es aquel cuyo punto b dista de 0,38197Δ de un extremo, y 0,61803Δ del otro extremo. Estos números están relacionados con la conocida razón o sección áurea, cuyas supuestas propiedades estéticas se remontan

hasta los antiguos Pitagóricos.

Resumen del método: Dados, en cada iteración, una tupla de puntos que sitúa el mínimo, el siguiente punto a probar es aquel que está en el mayor de los dos segmentos (a, b) o (b, c), a una distancia de 0.38197 veces la anchura de dicho segmento (medido desde la abscisa central de la tupla) ?. Si se empieza con una tupla cuyos segmentos no cumplen la razón áurea, el procedimiento de elegir puntos sucesivos en la media áurea del segmento mayor rápidamente convergerá a que los segmentos de la tupla final lo cumplan. La búsqueda mediante la razón áurea garantiza que cada nueva evaluación de la función sitúa el mínimo en un intervalo de tamaño 0.61803 veces el tamaño del intervalo precedente. Esto es comparable con el 0.5 del método de la bisección para encontrar raíces. Es un algoritmo seguro, pero lento.

El método programado en la rutina `mnbrak` sirve para situar el mínimo (Routine for Initially Bracketing a Minimum), previamente a la búsqueda del mismo.
ENTRADA: una función y distintos puntos iniciales: a, b, esta rutina busca el intervalo donde hay un mínimo en la dirección cuesta abajo entre $f(a)$ y $f(b)$.
SALIDA: tres nuevos puntos `ax`, `bx`, `cx` que delimitan el intervalo donde hay un mínimo de la función, y el valor de la función en estos 3 puntos.

La rutina `golden` realiza la búsqueda del mínimo por el método de la razón áurea antes descrito.
ENTRADA: una función unidimensional `f` y tres abscisas que delimitan el intervalo de un mínimo a, b, c (tales que b está entre a y c, y que $f(b)$ es menor que $f(a)$ y que $f(c)$).
SALIDA: la abscisa del mínimo (`xmin`) y el valor del mínimo (`fmin`).

Ambas rutinas admiten pasar parámetros directamente a la función `f` mediante la variable `varargin`, su último argumento de entrada.

C.2.2. Interpolación Parabólica y el método de Brent

El método de la razón áurea está pensado para manejar el peor caso de minimización de la función, en la que el mínimo es "escurridizo". Mas no hay porqué asumir lo peor, si la función es parabólica cerca del mínimo (caso genérico de una función suficientemente suave) entonces, la parábola ajustada a tres puntos cualesquiera nos indica en un sólo paso la posición del mínimo, o al menos nos sitúa muy cerca del mismo. Como se quiere encontrar una abscisa en lugar de una ordenada, el método se conoce técnicamente como interpolación parabólica inversa.

Descripción del método de Brent: en cualquier iteración se lleva cuenta de seis puntos (no necesariamente todos distintos), a, b, u, v, w y x, definidos de la siguiente forma: el mínimo está en el intervalo (a, b), x es el último punto con el menor valor de la función hasta el momento, w es el punto con el segundo valor más pequeño; v es el anterior valor de w; u es el punto de evaluación de la función más reciente. Dentro del algoritmo se considera el punto xm, el punto medio entre a y b, aunque aquí no se evalúe la función.

Se intenta la interpolación parabólica entre los puntos x, v, y w. Para que sea aceptada, la abscisa del mínimo de la parábola ajustada debe (i) estar en el intervalo (a, b), y (ii) suponer un desplazamiento desde el mejor valor actual x que sea menor que la mitad del paso anterior. Este segundo criterio asegura que los pasos parabólicos están convergiendo a algo, en lugar de rebotar en un ciclo no convergente. En el peor de los casos, en el que los pasos parabólicos son aceptados pero inútiles, el método aproximadamente alternará entre ajustes parabólicos y búsqueda mediante secciones áureas, convergiendo en virtud del último método.

La función `brent` implementa el método de Brent. Recibe y devuelve lo mismo que `golden`.

C.2.3. Utilizando la primera derivada

Aquí se pretende conseguir lo mismo que en los dos apartados anteriores (encontrar el mínimo de una función que está delimitado por una tupla de abscisas $[a, b, c]$), pero utilizando la capacidad adicional de evaluar la derivada de la función.

El esquema a seguir no es el que cabría esperar - mirar los puntos en los que la derivada se anula (condición de punto crítico)-, sino mantener la información de la tupla que sitúa el mínimo y utilizar la derivada para elegir nuevos puntos de prueba dentro del intervalo.

El signo de la derivada de la función en el punto central de la tupla $[a, b, c]$ indica si el siguiente punto de prueba se debe tomar en (a, b) o en (b, c). El valor de dicha derivada y el de la derivada en el segundo mejor punto indican el siguiente punto de prueba (método de la secante - interpolación lineal inversa), que es convergente (superlinealmente) de orden 1.618. Se imponen las mismas restricciones que en el método de Brent para este nuevo punto de prueba.

La función `dbrent` implementa el algoritmo de este apartado. Tiene casi la misma cabecera que `brent`, salvo que además necesita, como argumento de entrada, el nombre de la función que devuelve la derivada de la función a minimizar.

C.3. Métodos multidimensionales

C.3.1. Método del Simplex Cuesta Abajo

Se considera el tema de la minimización multidimensional, es decir, encontrar el mínimo de una función de varias variables independientes. Este apartado se sigue una estrategia distinta respecto de los dos siguientes: no utiliza una función que explícitamente realice minimizaciones unidimensionales, sino que se basa en un principio distinto, en el cual no figuran este tipo de minimizaciones en una dimensión.

El método del Simplex Cuesta abajo (Downhill Simplex Method) es idea de Nelder y Mead. El método sólo necesita evaluaciones de la función, no de la derivada. No es muy eficiente en términos del número de evaluaciones de la función que requiere. El método de Powell (§ C.3.2) es, casi con seguridad, más rápido en la mayoría de aplicaciones. Sin embargo, el método del Simplex Cuesta Abajo es frecuentemente el mejor método a utilizar si la figura de mérito es "conseguir algo que funcione rápidamente" para un problema computacionalmente no muy costoso.

El método tiene unos fundamentos geométricos que lo hacen atractivo de estudiar. Un *simplex* en un espacio $\mathbb{R}^M$ es una figura geométrica consistente en $M+1$ puntos (o vértices) y todos los segmentos y caras poligonales que los unen. En dos dimensiones, un simplex es un triángulo. En tres dimensiones, un simplex es un tetraedro, no necesariamente regular. En general, estamos interesados en simplexes no degenerados, es decir, que encierran un volumen interno finito $M-$dimesional. Si cualquier punto de un simplex no degenerado se toma como origen de coordenadas, los otros M puntos definen direcciones de vectores que son base del espacio $\mathbb{R}^M$.

La rutina `amoeba` implementa el nombrado algoritmo. Se debe pasar, además de la función a minimizar, un simplex inicial como argumento de entrada. La inicialización no consiste en un solo punto, sino en $M+1$. Si tenemos un punto $\mathbf{P}_0$ y queremos construir los otros M, se puede hacer así:

$$\mathbf{P}_i = \mathbf{P}_0 + \lambda \mathbf{e}_i \tag{C.1}$$

siendo $\mathbf{e}_i$ los vectores de la base canónica de $\mathbb{R}^M$, y λ es una constante que depende del factor de escala del problema considerado y estima el mismo.

A continuación, el método realiza una serie de pasos, la mayoría de los cuales mueven el punto del simplex con mayor valor de la función ("el punto más alto") a través de la cara opuesta del simplex hacia

un valor menor. Estos pasos se conocen como reflexiones y se construyen para conservar el volumen del simplex (mantenerlo no degenerado). Cuando puede, expande el simplex en una u otra dirección para tomar pasos más grandes. Cuando alcanza el comienzo de un valle o depresión, el método contrae el simplex en la dirección transversal y trata de ir valle abajo. Si hay una situación en la que el simplex esta tratando de "pasar a través del ojo de una aguja", se comprime en todas las direcciones alrededor del punto más bajo (mejor).

La función `amoeba` devuelve: una matriz `p` con el simplex final, el vector `y` con los valores de la función en los puntos del simplex final ($M+1$ filas de `p`), el número de evaluaciones de la función (`nfunk`). El valor del mínimo está en `y(1)` y la posición del mismo en la primera columna de `p` (`p(:,1)`), pues asume que se sigue el criterio habitual de vectores columna, en lugar de fila.

Criterios de terminación multidimensional

Determinar un criterio de terminación puede ser delicado en cualquier minimización multidimensional. Sin intervalos y con más de una variable independiente, ya no se puede aplicar el criterio de imponer cierta tolerancia a una sola de ellas.

A primera vista hay dos posibles lugares donde mirar un criterio de terminación: (i) en el espacio de partida $\mathbb{R}^M$ o (ii) en el cuerpo de llegada $\mathbb{R}$. En $\mathbb{R}^M$, se puede terminar cuando la distancia del desplazamiento en una iteración del algoritmo es menor, incrementalmente, que cierta tolerancia (`tol`). En $\mathbb{R}$, se podría exigir que el decremento en el valor de la función en el paso final sea menor que cierta tolerancia (`ftol`).

Mientras que `tol` no debería ser menor que la raíz cuadrada de la precisión de la computadora, es perfectamente correcto que `ftol` sea del orden de la precisión de la misma (o quizá un poco mayor para evitar errores de redondeo).

Cualquiera de los criterios anteriores falla por un único paso anómalo que, por una u otra razón, no lleva a ninguna parte. Entonces, es una buena idea reiniciar la rutina de minimización multidimensional desde el punto donde dice haber encontrado un mínimo.

En el método del Símplex Cuesta Abajo, se deben reinicializar M de los $M+1$ vértices del simplex mediante la ecuación (C.1), situando en el vértice $\mathbf{P}_0$ el mínimo actual. Reiniciar la búsqueda no debería ser costoso, pues el algoritmo una vez convergió a un mínimo y ahora arrancamos desde allí el algoritmo (punto de reinicio).

C.3.2. Métodos de un conjunto de direcciones

Sabemos cómo minimizar una función de una variable. Si empezamos en un punto $\mathbf{P}$ de $\mathbb{R}^M$ y procedemos en cierta dirección indicada por un vector $\mathbf{n}$, entonces cualquier función de M variables $f(\mathbf{P})$ puede ser minimizada a lo largo de la recta $\mathbf{n}$ por alguno de los métodos de minimización unidimensional. Se pueden pensar varios métodos de minimización multidimensional consistentes en sucesivas minimizaciones unidimensionales. Los distintos métodos se diferenciaran por cómo, en cada iteración, elijan la próxima dirección $\mathbf{n}$ de prueba. Todos estos métodos suponen la existencia de un sub-algoritmo en una "caja negra", que llamaremos `linmin`, cuya definición puede ser:

> Dados los vectores $\mathbf{P}, \mathbf{n}$ y una función f, encontrar el escalar λ que minimiza $f(\mathbf{P}+\lambda\mathbf{n})$. Reemplazar $\mathbf{P}$ por $\mathbf{P}+\lambda\mathbf{n}$. Reemplazar $\mathbf{n}$ por $\lambda\mathbf{n}$

Los métodos de este apartado y el siguiente siguen este esquema general de sucesivas minimizaciones a lo largo de una recta. En este apartado consideramos una clase de métodos cuya elección de las sucesivas direcciones no implica el cálculo explícito del gradiente de la función, en el siguiente apartado sí es necesario el cálculo de dicho gradiente. No es necesario especificar si `linmin` utiliza información

del gradiente o no. Esa elección depende de uno mismo y su optimización depende de la función en particular que tengamos entre manos. De todas formas, sería absurdo utilizar el gradiente en `linmin` y no hacerlo en la elección de las direcciones, ya que esto último puede reducir drásticamente el coste computacional total.

Pero si en nuestra aplicación concreta, el cálculo del gradiente está fuera de lugar, podríamos pensar en este sencillo método: tomar los vectores unitarios de la base canónica de $\mathbb{R}^M$, $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_M$ como un *conjunto de direcciones*. Utilizando `linmin`, moverse a lo largo de la primera dirección hasta su mínimo, desde allí, moverse a lo largo de la segunda dirección hasta su mínimo, y así sucesivamente, cíclicamente minimizando según las direcciones, hasta que que la función deja de decrecer.

Este sencillo método no es malo para muchas funciones. Incluso es más interesante el porqué es pésimo e ineficiente para determinadas funciones. En general, si las segundas derivadas de la función son mucho más grandes en magnitud en unas direcciones que en otras (valles muy estrechos), entonces serán necesarios muchos ciclos de minimizaciones a lo largo de los M vectores de la base.

Obviamente, lo que necesitamos es un conjunto de direcciones mejor que los $\mathbf{e}_i$. Todos los métodos que utilizan un *conjunto de direcciones* son recetas para actualizar ese conjunto según el método avanza, tratando de llegar a uno que (i) incluya varias direcciones muy buenas que nos llevarán lejos en valles estrechos, o incluso (ii) incluya un número de direcciones "no interferentes" con la propiedad especial de que la minimización realizada a lo largo de una de ellas no sea "estropeada" por subsiguientes minimizaciones a lo largo de otras, para evitar interminables ciclos de minimización a lo largo del conjunto de direcciones.

C.3.2.1. Direcciones conjugadas

Vale la pena explicitar matemáticamente este concepto de direcciones "no interferentes", convencionalmente llamadas direcciones conjugadas.

Primero, notar que si minimizamos una función a lo largo de una dirección $\mathbf{u}$, entonces, el gradiente de la función debe ser perpendicular a $\mathbf{u}$ en el mínimo; si no, todavía existirá una derivada direccional según $\mathbf{u}$ no nula.

A continuación tomar un punto $\mathbf{P}$ como origen de un sistema de coordenadas, cuyas coordenadas son $\mathbf{x}$. Entonces, cualquier función puede ser aproximada por su desarrollo en serie de Taylor

$$f(\mathbf{x}) = f(\mathbf{P}) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j + \cdots \approx c - \mathbf{b}^\top \mathbf{x} + \frac{1}{2} \mathbf{x}^\top \mathtt{A} \mathbf{x} \tag{C.2}$$

donde

$$c \equiv f(\mathbf{P}) \qquad \mathbf{b} = -\nabla f|_{\mathbf{P}} \qquad [\mathtt{A}]_{i,j} = \left. \frac{\partial^2 f}{\partial x_i \partial x_j} \right|_{\mathbf{P}} \tag{C.3}$$

La matriz $\mathtt{A}$ es el Hessiano de f particularizado en $\mathbf{P}$. En la aproximación (C.2), el gradiente se calcula fácilmente:

$$\nabla f = \mathtt{A}\mathbf{x} - \mathbf{b} \tag{C.4}$$

Esto implica que el gradiente se anulará (la función alcanzará un extremo) para el valor de $\mathbf{x}$ obtenido de resolver $\mathtt{A}\mathbf{x} = \mathbf{b}$.

¿Cómo cambia el gradiente ∇f según nos movemos en una dirección? Evidentemente,

$$\delta(\nabla f) = \mathtt{A}(\delta \mathbf{x}). \tag{C.5}$$

Supongamos que nos hemos movido en cierta dirección $\mathbf{u}$ hasta un mínimo y ahora nos proponemos mover según una nueva dirección $\mathbf{v}$. La condición de que el desplazamiento según $\mathbf{v}$ no estropee la

minimización según $\mathbf{u}$ es que el gradiente permanezca perpendicular a $\mathbf{u}$, esto es, que el cambio en el gradiente sea perpendicular a $\mathbf{u}$. Por la ecuación (C.5) esto se indica:

$$0 = \mathbf{u}^\top \delta(\nabla f) = \mathbf{u}^\top \mathtt{A}\mathbf{v} \tag{C.6}$$

Cuando se cumple (C.6) para dos vectores $\mathbf{u}$ y $\mathbf{v}$, se dice que son *conjugados*. Cuando la relación se cumple por parejas para todos los miembros de un conjunto de vectores, se dice que son un conjunto conjugado. Si se realizan sucesivas minimizaciones de una función a lo largo de un conjunto conjugado de direcciones, entonces no hace falta volver a minimizar en ninguna de dichas direcciones.

Un triunfo para un método de un conjunto de direcciones es encontrar un conjunto de M direcciones linealmente independientes y mutuamente conjugadas. Entonces, una pasada de M minimizaciones a lo largo de dichas direcciones llegará al mínimo de la forma cuadrática dada en (C.2). Para funciones f que no son exactamente cuadráticas, no se alcanzará el mínimo exactamente, pero repetidos ciclos de M minimizaciones convergerá, a su tiempo, cuadráticamente hacia el mínimo.

C.3.2.2. El método cuadráticamente convergente de Powell

Powell fue el primero que descubrió un método que produce M direcciones mutuamente conjugadas sin usar el Hessiano. Es el siguiente: Inicializar el conjunto de direcciones $\mathbf{u}_i$ a los vectores de la base:

$$\mathbf{u}_i = \mathbf{e}_i \qquad\qquad i = 1, \ldots, M \tag{C.7}$$

A continuación repetir la siguiente secuencia de pasos (procedimiento básico) hasta que la función deje de decrecer:

- Guardar el punto de comienzo como $\mathbf{P}_0$
- Para $i = 1, \ldots, M$, desplazar $\mathbf{P}_{i-1}$ al mínimo a lo largo de $\mathbf{u}_i$ y llamar a este punto $\mathbf{P}_i$
- Para $i = 1, \ldots, M-1$, asignar $\mathbf{u}_i \leftarrow \mathbf{u}_{i+1}$
- Asignar la nueva dirección $\mathbf{u}_M = (\mathbf{P}_M - \mathbf{P}_0)$
- Desplazar $\mathbf{P}_M$ al mínimo a lo largo de la dirección $\mathbf{u}_M$ y llamar a este punto $\mathbf{P}_0$

En 1964, Powell mostró que, para una forma cuadrática como (C.2), k iteraciones del anterior procedimiento básico produce un conjunto de direcciones $\mathbf{u}_i$ cuyos últimos k elementos son mutuamente conjugados. Por lo tanto, M iteraciones del procedimiento básico, sumando $M(M+1)$ minimizaciones unidimensionales, minimizarán exactamente una forma cuadrática.

Desafortunadamente existe un inconveniente en el algoritmo cuadráticamente convergente de Powell. El paso de despreciar $\mathbf{u}_1$ en favor de $\mathbf{P}_M - \mathbf{P}_0$ tiende a producir conjuntos de direcciones linealmente dependientes. Una vez que esto sucede, el procedimiento encuentra el mínimo de la función f sólo en un subespacio del ambiente $\mathbb{R}^M$, en otras palabras, proporciona una respuesta incorrecta. Por lo tanto no hay que utilizar el algoritmo tal y como está escrito.

Existen varias formas de resolver el problema de la dependencia lineal de las direcciones en el algoritmo de Powell, entre ellas:

1. Se puede reinicializar el conjunto de direcciones $\mathbf{u}_i$ a los vectores de la base $\mathbf{e}_i$ cada M o $M+1$ iteraciones del procedimiento básico. Este es un método útil, que es interesante si la función se asemeja a una forma cuadrática y se desea mucha exactitud.

2. Brent señala que una elección para reinicializar el conjunto de direcciones tan buena como los vectores de la base canónica es utilizar las columnas de cualquier matriz ortogonal. En lugar de desperdiciar la información de direcciones conjugadas conseguidas hasta el momento, él reinicializa el conjunto de direcciones a las direcciones principales de la matriz $\mathtt{A}$ (da un procedimiento para determinarlas). El cálculo se basa en la descomposición en valores singulares (SVD). Brent tiene otros trucos escondidos bajo su manga y su modificación del método de Powell es, probablemente, la mejor hasta la fecha. Desafortunadamente, es demasiado elaborada como para incluirla en [31].

3. Podemos abandonar la propiedad de convergencia cuadrática en favor de un esquema más heurístico (debido a Powell) que trata de encontrar unas pocas direcciones buenas en valles estrechos en lugar de M direcciones conjugadas. Este es el método que está implementado en la rutina `powell` y del que hablaremos un poco a continuación.

C.3.2.3. Descartando la dirección de máximo decrecimiento

Ahora que vamos a abandonar de la propiedad de convergencia cuadrática, ¿por qué es tan importante después de todo? Eso depende de la función que estemos minimizando. Algunas aplicaciones producen funciones con largos y retorcidos valles. La convergencia cuadrática no es particularmente ventajosa para un programa que debe explorar por un valle que se retuerce de una a otra dirección (¡hay M dimensiones!). A lo largo de la dirección más ancha del valle, un método cuadráticamente convergente está tratando de saltar al mínimo de una parábola que (todavía) no está allí, mientras que se estropea la conjugación de las $M-1$ direcciones transversales a causa de los giros y recovecos.

De todas formas, tarde o temprano llegamos a un mínimo aproximadamente elipsoidal (ecuación C.2 cuando $\mathbf{b}$, el gradiente, es cero). Entonces, dependiendo de cuanta exactitud queramos, un método con convergencia cuadrática nos puede ahorrar varias veces M^2 minimizaciones unidimensionales extra, ya que la convergencia cuadrática dobla el número de cifras en cada iteración.

La idea básica de la modificación del método de Powell de [31] es mantener $\mathbf{P}_M - \mathbf{P}_0$ como nueva dirección, ya que es, después de todo, la dirección media desplazada tras probar M posibles direcciones. Para valles cuya dirección ancha se está retorciendo lentamente, cabe esperar que la nueva dirección proporcione un gran desplazamiento a lo largo de la nueva dirección ancha. El cambio consiste en descartar la antigua dirección a lo largo de la cual la función f ha decrecido más. Esto parece paradójico, ya que esa dirección fue la *mejor* de la anterior iteración. Sin embargo, también tendrá una gran componente en la nueva dirección que se está añadiendo, así que su descarte nos da la mejor opción de evitar construir un conjunto linealmente dependiente.

Hay una serie de excepciones a esta idea básica. A veces es mejor no añadir ninguna dirección. Definamos

$$f_0 = f(\mathbf{P}_0) \qquad f_M = f(\mathbf{P}_M) \qquad f_E = f(2\mathbf{P}_M - \mathbf{P}_0)$$

Aquí f_E es el valor de la función más allá del punto "extrapolado" en la nueva dirección. Sea Δf la magnitud del mayor decrecimiento a lo largo de una dirección particular de la actual iteración del procedimiento básico (Δf es un número positivo). Entonces:

1. Si $f_E \geq f_0$, se debe mantener el antiguo conjunto de direcciones para la siguiente iteración, ya que dirección media $\mathbf{P}_M - \mathbf{P}_0$ está gastada, ha sido probada.

2. Si $2(f_0 - 2f_M + f_E)[(f_0 - f_M) - \Delta f]^2 \geq (f_0 - f_E)^2 \Delta f$, se debe mantener el antiguo conjunto de direcciones para la siguiente iteración, porque o bien (i) el decrecimiento a lo largo de la dirección media no era debido a ningún decrecimiento en una sola dirección, o bien (ii) hay una substancial segunda derivada a lo largo de la dirección media y parece que estamos cerca del fondo de su mínimo.

El fichero `powell.m` contiene la función principal `powell` y las subfunciones para la minimización unidimensional `linmin`.

`powell`
ENTRADA:

`P`	punto inicial de evaluación de la función: $\mathbf{P}_0 \in \mathbb{R}^M$
`func`	función $f : \mathbb{R}^M \to \mathbb{R}$ a minimizar
`varargin`	matriz `Xi` cuyas columnas son los vectores de una base de $\mathbb{R}^M$. Indican las primeras direcciones en que se minimiza la función, llamando al método `linmin`.

SALIDA:

`p`	el mejor vector de parámetros encontrado (posición del mínimo)
`fret`	Valor del mínimo
`xi`	el conjunto de direcciones actuales de evaluación
`iter`	número de iteraciones realizadas

Subfunción o rutina secundaria `linmin`
ENTRADA:

`P0`	punto en $\mathbb{R}^M$
`Xi`	dirección de $\mathbb{R}^M$ a lo largo de la cual minimizar
`func`	función $f : \mathbb{R}^M \to \mathbb{R}$ a minimizar

REALIZA: mueve y cambia `P0` por el punto donde `func` alcanza el mínimo según la dirección `Xi`, desde `P0`. Cambia `Xi` por el vector del desplazamiento que hemos hecho desde `P0`. Es decir, la misión de la "caja negra" que se indica al principio de (C.3.2). Para ello llama a las rutinas `mnbrak` y `golden`.

SALIDA:

`p`	parámetros donde se alcanza el mínimo en la dirección `Xi`
`xi`	desplazamiento en $\mathbb{R}^M$ desde el punto `P0` a `p`
`xmin`	abscisa del mínimo que devuelve `golden`
`fret`	valor del mínimo

A la hora de programar, quizá lo más interesante sea cómo construir en `linmin` la función unidimensional que hay que pasar a las funciones 1D (`f1dim` de [31]). Se hace con un poco de astucia gracias a los comandos `inline` y `sprintf`.

C.3.3. Métodos del gradiente conjugado

Consideremos el caso en el que somos capaces de calcular, en un punto $\mathbf{P} \in \mathbb{R}^M$ dado, no sólo el valor de la función $f(\mathbf{P})$, sino también el gradiente (vector de primeras derivadas parciales) $\nabla f|_{\mathbf{P}} \equiv \nabla f(\mathbf{P})$.

Utilizar el gradiente es ventajoso. Supongamos que la función puede ser aproximada por una forma cuadrática como en la ecuación (C.2).

$$f(\mathbf{x}) \approx c - \mathbf{b}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathtt{A}\mathbf{x} \tag{C.8}$$

El número de parámetros desconocidos de f es igual al número de parámetros libres de `A` y $\mathbf{b}$, que es $M(M+1)/2$, de orden M^2. Cambiar cualquiera de estos parámetros puede suponer un desplazamiento de la posición mínimo. Por lo tanto, no podemos esperar ser capaces de encontrar el mínimo hasta que hallamos recogido una cantidad de información equivalente, del orden de M^2 números.

En los métodos de un conjunto de direcciones de § C.3.2, se recoge la información necesaria haciendo del orden de M^2 minimizaciones unidimensionales separadas, cada una de las cuales requería "unas pocas" evaluaciones de la función. Ahora, cada evaluación del gradiente proporciona M nuevas componentes de información. Si las utilizamos sabiamente, sólo necesitaremos hacer del orden de M minimizaciones unidimensionales separadas. Es el caso de los algoritmos de esta sección.

No siempre implica una mejora por un factor de M en velocidad computacional. Como una burda aproximación, podríamos imaginar que el cálculo de cada componente del gradiente consume tanto tiempo como una evaluación de la función. En ese caso, habrá del orden de M^2 evaluaciones equivalentes de la función con y sin la información del gradiente. Aunque la mejora no sea de orden M, es aún así, considerable: (i) cada componente del gradiente típicamente ahorrará no sólo una evaluación de la función, sino varias, equivalente a decir una minimización unidimensional completa. (ii) Hay normalmente una alto grado de redundancia en las fórmulas para las distintas componentes del gradiente; cuando esto se cumple, especialmente cuando hay redundancia en la evaluación de la función,

entonces el cálculo del gradiente es significativamente menos costoso que M evaluaciones de la función.

Un error frecuente de principiante es suponer que cualquier manera razonable de incorporar la información del gradiente es tan buena como cualquier otra. Esta línea de pensamiento conduce al siguiente algoritmo (no muy bueno), llamado *the steepest descent method*, que se traduce como "método de la máxima pendiente".

> *Steepest Descent Method*: Empezar en un punto $\mathbf{P}_0$. Tantas veces como sea necesario, desplazarse del punto $\mathbf{P}_i$ al punto $\mathbf{P}_{i+1}$ mediante una minimización a lo largo de la recta que pasa por $\mathbf{P}_i$ y sigue la dirección cuesta abajo del gradiente local $-\nabla f|_{\mathbf{P}_i}$.

Existe un inconveniente con el método anterior (que se remonta a Cauchy). En un valle muy largo y achatado, el método realizará muchos pasos pequeños, incluso si el valle es una forma cuadrática perfecta.

Al igual que en la discusión que condujo a la fórmula (C.6), queremos una forma de proceder no hacia abajo según el nuevo gradiente, sino que preferimos una dirección que sea construida de tal forma que sea conjugada respecto del antiguo gradiente y, siempre que sea posible, respecto de todas las direcciones cruzadas. Los métodos que versan sobre esta construcción se llaman *métodos del gradiente conjugado*.

Comenzando con un vector inicial $\mathbf{g}_0$ y haciendo $\mathbf{h}_0 = \mathbf{g}_0$, el método del gradiente conjugado construye dos secuencias de vectores a partir de las ecuaciones recurrentes:

$$\mathbf{g}_{i+1} = \mathbf{g}_i - \lambda_i \mathtt{A}\mathbf{h}_i \qquad \mathbf{h}_{i+1} = \mathbf{g}_{i+1} + \gamma_i \mathbf{h}_i \qquad i = 0, 1, 2, \ldots \tag{C.9}$$

Los vectores satisfacen las condiciones de ortogonalidad y conjugación:

$$\mathbf{g}_i^\top \mathbf{g}_j = 0 \qquad \mathbf{h}_i^\top \mathtt{A}\mathbf{h}_j = 0 \qquad \mathbf{g}_i = \mathbf{h}_j = 0 \qquad j < i \tag{C.10}$$

Los escalares λ_i y γ_i vienen dados por las fórmulas:

$$\lambda_i = \frac{\mathbf{g}_i^\top \mathbf{g}_i}{\mathbf{h}_i^\top \mathtt{A}\mathbf{h}_i} = \frac{\mathbf{g}_i^\top \mathbf{h}_i}{\mathbf{h}_i^\top \mathtt{A}\mathbf{h}_i} \tag{C.11}$$

$$\gamma_i = \frac{\mathbf{g}_{i+1}^\top \mathbf{g}_{i+1}}{\mathbf{g}_i^\top \mathbf{g}_i} \tag{C.12}$$

Se supone que el Hessiano o matriz Hessiana `A` es simétrico. Supongamos que conocemos `A` en (C.8), entonces podríamos utilizar la construcción (C.9) para encontrar sucesivamente direcciones conjugadas $\mathbf{h}_i$ a lo largo de las cuales minimizar. Después de M de éstas, habremos eficientemente alcanzado el mínimo de la forma cuadrática. Mas desconocemos `A`. He aquí un notable teorema que viene en nuestra ayuda:

Teorema 1 *Supongamos que $\boldsymbol{g}_i = -\nabla f|_{\boldsymbol{P}_i}$, para cierto punto $\boldsymbol{P}_i$, donde f es de la forma (C.8). Supongamos que se procede desde $\boldsymbol{P}_i$ a lo largo de la dirección $\boldsymbol{h}_i$ hasta el mínimo local de f situado en algún punto $\boldsymbol{P}_{i+1}$ y hágase $\boldsymbol{g}_{i+1} = -\nabla f|_{\boldsymbol{P}_{i+1}}$. Entonces, este $\boldsymbol{g}_{i+1}$ es el mismo vector que se hubiera construido según (C.9). (¡Y se ha construido sin conocimiento de `A`!).*

Tenemos, pues, la base de un algoritmo que no requiere el conocimiento del Hessiano `A` ni de la memoria necesaria para almacenar dicha matriz. Una secuencia de direcciones $\mathbf{h}_i$ es construida, utilizando sólo minimizaciones 1D, evaluaciones del gradiente de la función y un vector auxiliar para almacenar el último de la secuencia de vectores $\mathbf{g}$.

El algoritmo descrito hasta ahora es la versión original de Fletcher-Reeves del algoritmo del gradiente conjugado. Más tarde, Polak y Ribiere introdujeron un pequeño, pero a veces significativo, cambio. Propusieron utilizar la forma:

$$\gamma_i = \frac{(\mathbf{g}_{i+1} - \mathbf{g}_i)^\top \mathbf{g}_{i+1}}{\mathbf{g}_i^\top \mathbf{g}_i} \tag{C.13}$$

en lugar de la ecuación (C.12). Ambas γ_i son iguales por las condiciones de ortogonalidad (C.10). Son iguales para formas cuadráticas exactas. En el mundo real, sin embargo, la función normalmente no es exactamente una forma cuadrática: a pesar de llegar al mínimo de una forma cuadrática, todavía habrá que seguir iterando. La fórmula de Polak y Ribiere consigue una mejor transición hacia estas iteraciones: cuando la búsqueda no mejora tiende a reiniciar el procedimiento del gradiente conjugado.

La rutina `frprmn` implementa la variante de Polak-Ribiere, que es la recomendada en [31], mas con sólo cambiar una línea de código cambia a la variante Fletcher-Reeves. Permite elegir la forma de calcular el gradiente particularizado en el punto pasado a la rutina (`p`): (i) mediante una función que le pasemos explícitamente, `dfunc(p)` (contenida en `varargin`) o (ii) numéricamente, mediante la rutina `fdjac` en caso de que `varargin` sea vacío.

ENTRADA:

`p`	punto en $\mathbb{R}^M$
`func`	función $f : \mathbb{R}^M \to \mathbb{R}$ a minimizar
`varargin`	nombre de la rutina para calcular el gradiente

SALIDA:

`p`	parámetros donde se alcanza el mínimo en la dirección `Xi`
`fret`	valor del mínimo
`iter`	número de iteraciones realizadas

El fichero `frprmn.m` tiene varias funciones locales (sólo accesibles desde la rutina principal del mismo nombre - `frprmn`): `dlinmin` y `linmin`. A veces llamo a las funciones locales "subfunciones" o "rutinas secundarias".

La función local `dlinmin` realiza las minimizaciones 1D en el caso (i) de pasar a `frprmn` el nombre de la rutina que devuelve el gradiente, y emplea información del mismo para realizar las minimizaciones, es decir, llama a la rutina `dbrent` en lugar de `golden` o `brent`. Si no se indica nombre de la rutina que devuelve el gradiente (caso (ii)), las minimizaciones 1D se hacen sin información del mismo (llamando a `golden` o `brent`).

C.3.4. Método de enfriamiento simulado

El método enfriamiento simulado (Simulated Annealing) es una técnica que ha atraído mucha atención porque es adecuado para grandes problemas de optimización, especialmente aquellos en los que un mínimo global está escondido entre muchos mínimos locales de mayor valor. A efectos prácticos, el enfriamiento simulado "ha resuelto" el famoso problema del vendedor ambulante (*travelling salesman*). También se aplica con éxito para diseñar complejos circuitos integrados. Las dos aplicaciones anteriores son ejemplos de *minimización combinatoria*. Hay una función objetivo a minimizar, como siempre; pero el espacio sobre el cual está definida la función no es el sencillo espacio de parámetros de variación continua, sino más bien un espacio discreto muy grande con todas las posibles combinaciones de posiciones de circuitos u orden de las ciudades.

Sin embargo, también se puede aplicar a espacios de parámetros de variación continua, como venimos haciendo hasta ahora. Como veremos, el enfriamiento simulado prueba pasos aleatorios; mas en un valle muy estrecho y largo, casi todos esos pasos son cuesta arriba. Es necesario explicarlo un poco más.

En el corazón del método del enfriamiento simulado existe un paralelismo con la termodinámica, específicamente con la manera en que los líquidos se congelan y cristalizan, o los metales se enfrían. A altas temperaturas, las moléculas de un líquido se mueven libremente. Si el líquido es enfriado lentamente, se pierde la movilidad o agitación térmica. Los átomos suelen ser capaces de alinearse entre sí y formar un cristal puro que está completamente ordenado en cualquier dirección hasta distancias de un billón de veces el tamaño de un átomo. El cristal es el estado de mínima energía del sistema. El hecho sorprendente es que para sistemas lentamente enfriados, la naturaleza es capaz de encontrar

este estado de mínima energía. De hecho, si un metal líquido es enfriado rápidamente, no alcanza este estado, sino que termina en un estado policristalino o amorfo, de mayor energía.

Así que la esencia del proceso es enfriar lentamente, permitiendo tiempo suficiente para que los átomos se redistribuyan según pierden movilidad. Esta es la definición técnica de enfriamiento (*annealing*), y es esencial para asegurar que se alcanza un estado de baja energía.

Aunque la analogía no es perfecta, en cierto sentido todos los algoritmos de minimización vistos en este capítulo se corresponden con enfriamiento rápido. En todos los casos hemos tratado de llegar rápidamente a la solución: desde el punto inicial vamos inmediatamente cuesta abajo tanto como podamos. Esto conduce, habitualmente a un mínimo local, no a uno global. El propio algoritmo de minimización de la naturaleza está basado en un procedimiento bastante diferente. La conocida función densidad de probabilidad (f.d.p.) de Boltzmann,

$$\text{Prob}(E) \sim \exp(-E/kT)$$

expresa la idea de que un sistema en equilibrio térmico a temperatura T tiene su energía distribuida probabilísticamente distribuida entre todos los posibles estados E. Incluso a baja temperatura, existe una posibilidad, aunque muy pequeña, de que un sistema está en un estado de alta energía. Por lo tanto, existe la correspondiente posibilidad de que un sistema salga de un mínimo local de energía en favor de encontrar uno mejor, global. La cantidad k (constante de Boltzmann) es una constante de la naturaleza que relaciona temperatura y energía. En otras palabras, el sistema a veces va cuesta arriba así como cuesta abajo; pero cuanto menor es la temperatura, menos probable es cualquier excursión cuesta arriba significativa.

En 1953, Metropolis y colaboradores fueron los que primero incorporaron estos principios en cálculo numérico. Dadas varios posibles cambios, supusieron que un sistema termodinámico simulado cambiaba su configuración de la energía E_1 a la energía E_2 con probabilidad $p = \exp[-(E_2 - E_1)/kT]$. Nótese que si $E_2 < E_1$, esta probabilidad es mayor que la unidad; en tales casos, se asigna al cambio un probabilidad $p = 1$, es decir, el sistema siempre cambia. Este esquema general, de siempre tomar un paso cuesta abajo y *de vez en cuando* uno cuesta arriba, se conoce como el algoritmo de Metropolis.

Para utilizar el algoritmo de Metropolis para sistemas distintos de los termodinámicos se deben proporcionar los siguientes elementos:

1. Una descripción de las posibles configuraciones del sistema.

2. Un generador de cambios aleatorios en las configuraciones; estos cambios son las opciones presentadas al sistema.

3. Una función objetivo E (análoga a la energía) cuya minimización es la meta del procedimiento.

4. Un parámetro de control T (análogo a la temperatura) y un esquema de enfriamiento que indica cómo disminuye, es decir, después de cuántos cambios aleatorios en la configuración es T decrementado y en qué cantidad o proporción. En este contexto, el significado de "alto" o "bajo" T y la asignación de un esquema de enfriamiento suele requerir del método de ensayo y error.

C.3.4.1. Minimización continua por enfriamiento simulado

Las ideas básicas del enfriamiento simulado son aplicables a problemas de optimización con espacios M-dimensionales de parámetros de control de variación continua, esto es, encontrar el mínimo (global) de una función $f(\mathbf{x})$, en presencia de muchos mínimos locales, siendo $\mathbf{x} \in \mathbb{R}^M$. Los cuatro elementos del algoritmo de Metropolis son los siguientes: la función objetivo f; el estado del sistema es $\mathbf{x}$; el parámetro de control es algo parecido a una temperatura, con un esquema de enfriamiento que la va disminuyendo gradualmente; y debe existir un generador de cambios aleatorios en la configuración, es

decir, un procedimiento para tomar un paso aleatorio desde $\mathbf{x}$ hasta $\mathbf{x} + \Delta\mathbf{x}$.

El último de estos elementos es el más problemático. La literatura hasta la fecha describe diferentes esquemas de elegir $\Delta\mathbf{x}$, ninguno de los cuales inspira completa confianza a los autores de [31]. Es un problema de eficiencia: un generador de cambios aleatorios es ineficiente cuando, existiendo un paso cuesta abajo, él a pesar de ello casi siempre propone un paso cuesta arriba. Un buen generador no debería volverse ineficiente en valles estrechos, ni tampoco debería volverse más y más ineficiente a medida que se acerca la convergencia a un mínimo.

En [31], la manera de implementar el enfriamiento simulado sobre espacios de parámetros continuos es utilizar una modificación del método del Simplex Cuesta Abajo (§ C.3.1). Esto implica cambiar la descripción del estado del sistema: en lugar de un punto $\mathbf{x}$, un simplex de $M + 1$ puntos. Los pasos o movimientos son los mismos que los descritos en § C.3.1, o sea, reflexiones, expansiones y contracciones del simplex. La implementación del algoritmo de Metropolis es un poco sutil: se suma una variable aleatoria positiva, logarítmicamente distribuida, proporcional a la temperatura T, al valor de la función asociado a cada vértice del simplex, y se resta una una variable aleatoria similar al valor de la función de cada nuevo punto que es probado como sustituto de un vértice. Al igual que el procedimiento normal de Metropolis, este método siempre acepta un verdadero paso cuesta abajo, pero a veces acepta uno cuesta arriba. En el límite $T \longrightarrow 0$, este algoritmo se reduce al método del simplex cuesta abajo y converge a un mínimo local.

Para un valor finito de T, el simplex se expande a una escala que aproxima el tamaño de la región que es posible alcanzar a esta temperatura, y entonces ejecuta un movimiento estocástico, browniano dentro de esa región, probando nuevos y aproximadamente aleatorios parámetros. La eficiencia con que una región es explorada no depende de su anchura u orientación. Si se reduce la temperatura con suficiente lentitud, parece bastante probable que el simplex se contraiga en la región que contiene el menor mínimo relativo encontrado.

El esquema de enfriamiento programado consiste en reducir T a $(1 - \epsilon)T$ cada m pasos. El generador de números aleatorios es `rand`, de MATLAB.

El fichero `amebsa.m` contiene la rutina principal `amebsa` y la rutina local `amotsa`. Veamos la rutina principal:

ENTRADA:

`P`	matriz del simplex inicial
`Y`	vector con los valores de la función en los puntos del simplex inicial
`Pb,Yb`	mejor punto y su ordenada encontrados. Aunque en realidad es lo que devuelve el método, se utiliza para no perder el mínimo de una iteración a otra.
`ftol`	tolerancia que imponemos
`funk`	función a minimizar
`temptr`	temperatura de control
`Iter`	el número de iteraciones que realizamos a esa temperatura

REALIZA: `Iter` evaluaciones de la función a la temperatura `temptr`, después termina. Una vez hecho esto, desde fuera (`xamebsa`) se debe disminuir `temptr` de acuerdo con el esquema de enfriamiento, reiniciar `iter` y volver a llamar a la rutina (dejando el resto de parámetros inalterados entre llamadas). Si `iter` es devuelta con un valor positivo, entonces significa que se ha alcanzado la convergencia de la tolerancia impuesta. Si se inicializa `yb` a un valor muy grande en la primera llamada, entonces `yb` y `pb` devolverán el mejor valor de la función y posición encontrados hasta el momento (incluso si no es un punto del simplex).

SALIDA:

`p`	matriz del simplex final
`y`	vector con los valores de la función en los puntos del simplex final ($M+1$ columnas de `p`)
`pb,yb`	posición del mínimo encontrado y su ordenada.
`iter`	parámetro de control de la rutina `xamebsa`

La rutina `xamebsa` está sacada del libro de ejercicios del Numerical Recipes in C y realiza las veces de encapsulado: es a la que hay que llamar si se dispone de la función a minimizar y el punto inicial.

ENTRADA:

`tfunk`	función a minimizar $f : \mathbb{R}^M \to \mathbb{R}$
`xoff`	parámetros iniciales de búsqueda (punto inicial $\mathbf{P}_0 \in \mathbb{R}^M$)
`temptr`	temperatura inicial T. La temperatura disminuye en cada iteración
`iiter`	numero de iteraciones a cada valor de la temperatura

En el libro de ejercicios de [31] se realizan varias pruebas para la determinación de los valores de T e `Iter`. Un valor bajo de temperatura ($T = 10^3$) e `iter` $= 20$ puede no ser suficiente para encontrar un mínimo global si el punto inicial (o semilla) está situado en un mínimo local abrupto. En cambio, para una temperatura de $T = 10^6$ y 10 o 20 evaluaciones por temperatura casi siempre se alcanza el mínimo global independientemente de la semilla.

Si se realizan 2 evaluaciones de la función a cada temperatura, se está implementando un enfriamiento rápido y dependiendo de la semilla, puede que se alcance el mínimo global o no. Hay zonas de convergencia asegurada y otras en las que no sucede así.

REALIZA: Implementa el esquema de enfriamiento exponencial de razón $(1-\epsilon)$, inicializa el simplex a partir del punto inicial de búsqueda y para cada valor de temperatura T llama a `amebsa`, informa de la iteración en curso, la temperatura actual, el mínimo valor hasta el momento y el vector de parámetros donde se alcanza.

En el libro de ejercicios se utiliza $\epsilon = 0{,}2$, es decir, que la temperatura se reduce en un 20 % de una iteración a otra.

SALIDA:

`p`	Simplex final
`y`	valores de la función en los vértices del simplex
`pb`	posición del mínimo
`ybb`	valor del mínimo
`nit`	número de iteraciones realizadas

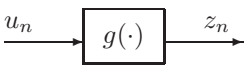
La temperatura indica la media de la fluctuación térmica que se suma o se resta a los valores de la función en cada vértice del simplex. Esta fluctuación z sigue la distribución de una variable aleatoria (v.a.) exponencial y es creada a partir de una v.a. uniforme u.

Hay dos formas de ver la relación de u y z: el problema directo y el problema inverso.

1. Problema inverso: (el de la persona que analiza la rutina que le dan y la intenta comprender)

   Dada la transformación de v.a. $z = -T\ln(u)$, y sabiendo que u son muestras de una v.a. uniforme en $(0,1)$, ¿Qué distribución sigue la v.a. z y cuál es su media?

   Modelamos el problema con una transformación no lineal: $z = g(u) = -T\ln(u)$ por la que entran las muestras u_n de una v.a. uniforme y salen las muestras z_n de otra v.a., la cual queremos caracterizar.



La v.a. uniforme tiene una función densidad de probabilidad (f.d.p.)

$$f_U(x) = \begin{cases} 1 & 0 < x < 1 \\ 0 & \text{resto} \end{cases}$$

Para, a partir de esta, hallar la f.d.p. de la v.a. z necesitamos un Teorema Fundamental:

Teorema 2 *Teorema Fundamental de cálculo de la f.d.p. de una transformación de v.a.*

Este teorema se puede aplicar si U es una v.a. continua (no puede ser discreta o mixta); la trasformación $g(u)$ es continua, derivable y no constante en el dominio de U, Ω_U. Llamamos u_i a las raíces solución de la ecuación $z = g(u)$, $i = 1, \ldots,$ numero de raices.

Siendo g' la derivada de g, entonces:

$$f_Z(z) = \sum_{\forall u_i} \frac{f_U(u_i)}{|g'(u_i)|}$$

Veamos que es lícito aplicar el teorema, ya que se cumplen los requisitos: una v.a. uniforme U tiene una f.d.p. continua; la función $g(u) = -T\ln(u)$ es continua y no constante en $\Omega_U = (0,1)$, con derivada $g'(u) = -T/u$. La ecuación $z = -T\ln(u)$ sólo tiene una raíz, $u_1 = u = e^{-z/T}$.

$$f_Z(z) = \sum_{\forall u_i} \frac{f_U(u_i)}{|g'(u_i)|} = \frac{f_U(e^{-z/T})}{|g'(e^{-z/T})|} = \frac{f_U(e^{-z/T})}{\frac{|-T|}{|e^{-z/T}|}} = \frac{1}{T} e^{-z/T} f_U(e^{-z/T})$$

Y $f_U(e^{-z/T})$ es un pulso de altura unidad en $z \in (0,\infty) = \Omega_Z$, por lo que

$$f_Z(z) = \begin{cases} \frac{1}{T} e^{-z/T} & z > 0 \\ 0 & \text{resto} \end{cases}$$

Observamos que z *es* una v.a. exponencial de parámetro $a = \frac{1}{T}$:

$$f_e(z) = ae^{-az} \qquad a > 0,\ z > 0$$

La media de una v.a. de este estilo es bien conocida:

$$\mathrm{E}[Z] = \frac{1}{a} = T$$

Hemos llegado a la interpretación del parámetro temperatura, T: la media de la fluctuación térmica que se suma o se resta a los valores de la función f en cada vértice del simplex para explorar aleatoriamente el espacio de parámetros, $\mathbb{R}^M$.

2. Problema directo: (el del programador de la rutina)

   Si dispongo de un generador de números aleatorios, `rand`*, que genera muestras de un v.a. uniforme en $(0,1)$ y quiero generar muestras de una v.a. exponencial de media T, ¿Cómo lo hago?*

   La incógnita, en este caso, es la transformación que hace que las muestras de una v.a. uniforme se conviertan en muestras de una v.a. exponencial.

   El método de la transformación inversa sirve para generar v.a. distintas de la uniforme, a partir de ésta. Se hace en dos pasos:

   a) Generar un número aleatorio u de una v.a. uniforme en $(0,1)$.

   b) Generar la v.a. como $z = F_Z^{-1}(u)$, donde $F_Z(z)$ representa la función de distribución acumulada de la v.a. a generar.

Ejemplo: v.a. exponencial;

$$f_Z(x) = ae^{-az} \;\Rightarrow\; F_Z(z) = 1 - e^{-az} \;\Rightarrow\; F_Z^{-1}(z) = -\frac{1}{a}\ln(1 - F_Z(z))$$

El algoritmo de generación según el método de la función inversa queda como:

a) Generar un número aleatorio u de una v.a. uniforme en $(0, 1)$.

b) $z = -\frac{1}{a}\ln(1-u) = -T\ln(u)$.

En el último paso se ha sustituido a y se ha empleado la observación de que el numero $(1-u)$ sigue también una distribución uniforme en $(0, 1)$.

Apéndice D

Manual de referencia

Este capítulo recoge las principales especificaciones de las rutinas *más importantes* programadas en MATLAB para el proyecto, salvo las comentadas en capítulo aparte sobre optimización según [31]. Los algoritmos ya han sido descritos en capítulos anteriores, ahora sólo se hará referencia a ellos y se explicarán algunos detalles de la implementación.

Algunas no son específicas de la visión artificial, sino que son rutinas de manipulación de matrices, etc. Hay ciertas rutinas que tienen su equivalente o rutina similar dentro de las programadas en el proyecto ADREP-3D (TIC2001−3069), ya que fueron programadas antes de revisar las rutinas proporcionadas o simplemente porque la acción de programarlas era un ejercicio instructivo.

D.1. Optimización de Levenberg-Marquardt

La presente sección está, en su mayoría, basada en el apéndice 4 de [1]. He programado distintas versiones del algoritmo LM, según iba leyendo el citado apéndice. Los algoritmos están descritos en orden creciente de complejidad debida a la explotación de características propias del enunciado del problema que se pretende resolver: el marco descriptivo común § 4.3. La nomenclatura seguida para nombrar las funciones de MATLAB que implementan los algoritmos es la siguiente:

Letra	Significado
`LM`	Levenberg-Marquardt
`B`	Básico
`P`	Particionado o Proyección
`S`	Disperso (Sparse)
`I`	Supone matriz de covarianzas $\Sigma_{\mathbf{X}} = \mathrm{Id}$
`H`	Matriz de la Homografía
`F`	Matriz Fundamental
`T`	Tensor Trifocal
`BA`	Ajuste de Haces (Bundle Adjustment)

Podemos clasificar los algoritmos, a parte de la complejidad, por su propósito: si es de propósito general o es específico de una aplicación. Siguiendo la notación, los algoritmos ordenados y de propósito general son:

`LMB`	Algoritmo LM **B**ásico
`LMPB`	Algoritmo LM **P**articionado **B**ásico
`LMPSI`	Algoritmo LM **P**articionado **S**parse (Disperso) [suponiendo: $\Sigma_{\mathbf{X}} = \mathrm{Id}$]
`LMPSI_BA`	Algoritmo LM aplicado al ajuste de haces en general [suponiendo: $\Sigma_{\mathbf{X}} = \mathrm{Id}$]
`LMSI`	Algoritmo LM **S**parse (Disperso) [suponiendo: $\Sigma_{\mathbf{X}} = \mathrm{Id}$]

Se podría llamar al algoritmo `LMPSI_BA` con el nombre de algoritmo particionado y doblemente disper-

so, sin embargo se le ha puesto el apellido de la aplicación general para la que se utiliza. Los algoritmos aplicados y ordenados son los de la siguiente tabla:

`LMBPI`	Algoritmo `LMB` aplicado al cálculo de la matriz de proyección
`LMPSHI_afin`	Algoritmo `LMPSI` aplicado al cálculo de homografías 2D, `H`
`LMPSFI`	Algoritmo `LMPSI` aplicado al cálculo de la matriz Fundamental, `F`
`LMPSTI`	Algoritmo `LMPSI` aplicado al cálculo del Tensor Trifocal, $\mathcal{T}$
`LMPSBAI`	Algoritmo LM aplicado al ajuste de haces proyectivo
`LMPSBAI_E`	Algoritmo LM aplicado al ajuste de haces Euclídeo

Todos los algoritmos anteriores necesitan conocer la función modelo $f : \mathbf{P} \rightarrow \widehat{\mathbf{X}}$. No es necesario pasarles la función de coste, porque es inmediata si se conoce f. Basándome en [31], programé otro algoritmo LM, `LMBchi2` , pero que sólo necesita conocer la función de coste, no el modelo (al igual que la rutina `fminunc`, propia de MATLAB). Esta rutina supone que la función cerca del mínimo se puede aproximar mediante los dos primeros términos del desarrollo en serie de Taylor, por lo que se necesita una rutina de propósito general que calcule el Hessiano, como puede ser `hessiano` o `hessianofd`.

D.1.1. Algoritmo básico según la función modelo

En esta sección se detalla el algoritmo de Levenberg-Marquardt Básico, según [1, pág. 569].

DADOS un vector de medidas $\mathbf{X}$ con matriz de covarianzas $\Sigma_{\mathbf{X}}$, una estimación inicial del vector de parámetros $\mathbf{P}$ y una función: $f : \mathbf{P} \rightarrow \widehat{\mathbf{X}}$ que transforma el vector de parámetros en una estimación del vector de medidas.

OBJETIVO Encontrar el vector de parámetros $\mathbf{P}$ que minimiza $\|\epsilon\|^2_{\Sigma_{\mathbf{X}}} = \epsilon^\top \Sigma_{\mathbf{X}}^{-1} \epsilon$ donde $\epsilon = \mathbf{X} - \widehat{\mathbf{X}}$.

ALGORITMO

(i) Inicializar una constante $\lambda = 0{,}001$ (valor típico) y calcular $\Sigma_{\mathbf{X}}^{-1}$.

(ii) Calcular la matriz jacobiana $\mathtt{J}$, el vector de error $\epsilon = \mathbf{X} - f(\mathbf{P})$ y el coste $\|\epsilon\|^2_{\Sigma_{\mathbf{X}}}$.

(iii) Crear las ecuaciones normales $\mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \mathtt{J} \delta = \mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \epsilon$, es decir, $\mathtt{N}\delta = \varepsilon$

(iv) Aumentar las ecuaciones normales: $\mathtt{N}^* \delta = \varepsilon$

(v) Resolver $\delta = (\mathtt{N}^*)^{-1} \varepsilon$

(vi) Calcular el nuevo vector de error $\epsilon_{\mathbf{P}+\delta}$ y su coste $\|\epsilon_{\mathbf{P}+\delta}\|^2_{\Sigma_{\mathbf{X}}}$.

(vii) Si el nuevo coste es menor que el antiguo, $\|\epsilon_{\mathbf{P}+\delta}\|^2_{\Sigma_{\mathbf{X}}} < \|\epsilon\|^2_{\Sigma_{\mathbf{X}}}$, se actualiza el vector de parámetros $\mathbf{P} = \mathbf{P} + \delta$, se disminuye λ en un factor de 10 y se empieza de nuevo en (ii) o se termina.

(viii) Si el nuevo coste es mayor que el antiguo, $\|\epsilon_{\mathbf{P}+\delta}\|^2_{\Sigma_{\mathbf{X}}} \geq \|\epsilon\|^2_{\Sigma_{\mathbf{X}}}$, se utiliza el antiguo $\mathbf{P}$, se incrementa λ en un factor de 10 y se vuelve a (iv).

La condición de terminación que he puesto es dejar de iterar en la tercera ocasión seguida que el coste decrece en una cantidad insignificante, es decir, si se cumplen tres veces seguidas el incremento fraccional es menor que 10^{-4}.

$$\left| 1 - \frac{\text{coste}_{\text{new}}}{\text{coste}_{\text{old}} + 10^{-10}} \right| < \text{tol} = 10^{-4}$$

El término 10^{-10} del denominador está para evitar indeterminaciones si el coste es todavía más pequeño. $\text{coste}_{\text{old}}$ es el mínimo coste hasta la nueva evaluación y no se permiten más de `MaxNumiter` iteraciones (normalmente 100) en caso de que sea una condición difícil de ser satisfecha.

Otra tendencia que puede suceder es que el algoritmo llegue a un mínimo y no consiga alcanzar la tolerancia fraccional impuesta, en esta situación el parámetro λ cada vez se hace más grande y ya nunca vuelve a decrecer. Para evitar esta situación, se detiene el algoritmo cuando $\lambda \geq \lambda_{\text{máx}}$, típicamente $\lambda_{\text{máx}} = 10^5$.

La rutina `LMB` es la implementación del algoritmo descrito. Su entrada-salida es la siguiente:

ENTRADA:

`X`	vector de medidas u observaciones objetivo
`Ex`	(opcional) matriz de covarianzas del vector de medidas
`P0`	estimación inicial del vector de parámetros $\mathbf{P}$
`func`	nombre de la función modelo $f : \mathbf{P} \mapsto \widehat{\mathbf{X}}$
`varargin`	argumentos opcionales que necesite la función `func`. `func(P,varargin:)`;

SALIDA:

`P0`	vector de parámetros del mínimo
`coste_min`	valor del mínimo (de la función de coste)
`X_min`	valor del vector de medidas en el mínimo
`Ep`	(opcional) matriz de covarianzas de los parámetros estimados (`P0`)

Si al llamar a la rutina se pide el último argumento de salida, se calcula la matriz de covarianzasdel mejor vector de parámetros $\Sigma_{\mathbf{P}}$, en caso contrario no se gasta esfuerzo en realizar dicha tarea innecesariamente. Los pasos son:

(i) Hallar la matriz jacobiana `J`, particularizada en el mejor $\mathbf{P}$
(ii) Calcular $\Sigma_{\mathbf{P}} = (\mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \mathtt{J})^+$

No es necesario proporcionar una rutina para la función de coste puesto que es conocida dentro de la propia rutina de minimización:

$$\|\mathbf{X} - f(\mathbf{P})\|^2_{\Sigma_{\mathbf{X}}} = \|\epsilon\|^2_{\Sigma_{\mathbf{X}}} = \epsilon^\top \Sigma_{\mathbf{X}}^{-1} \epsilon$$

También lleva otro atajo interno: si al llamar a la rutina no se proporciona la matriz de covarianzas en `Ex`, o se pasa la matriz identidad, la rutina se da cuenta y no calcula la inversa de la matriz de covarianzas identidad ni tampoco realiza operaciones innecesarias de multiplicación por dicha matriz al calcular el coste.

Además, la rutina permite optimizar sólo respecto a ciertos parámetros de la función. Para indicárselo, el argumento de entrada `P0` debe tener dos columnas: la primera es la estimación inicial para la búsqueda y la segunda es un vector binario cuyos elementos con valor 1 indican los parámetros respecto de los cuales optimizar, manteniendo fijo el resto. Esta opción sólo es posible en esta rutina y en `LMSI`.

En la figura D.1 se intenta ilustrar el funcionamiento del algoritmo de optimización en el caso particular de que la matriz de covarianzas $\Sigma_{\mathbf{X}}$ sea la identidad. Una justificación de porqué utilizar un parámetro interno de control λ se realizará en la próxima sección, § D.1.2.

D.1.2. Algoritmo básico según la función de coste

Aún en el caso de que una función de coste no admita una expresión en términos de una función modelo f es posible desarrollar el algoritmo LM. La próxima rutina está basada en [31], sección 15.4, y supone que la función (de coste) $g : \mathbb{R}^M \to \mathbb{R}$ cuyo mínimo estamos buscando admite, cerca del mínimo, una aproximación cuadrática mediante los dos primeros términos del desarrollo en serie de Taylor. Tal desarrollo alrededor del punto $\mathbf{P}$ es:

$$g(\mathbf{P} + \delta) \approx g(\mathbf{P}) + \delta^\top \nabla g|_{\mathbf{P}} + \frac{1}{2}\delta^\top \mathtt{H} \delta \tag{D.1}$$

siendo $\nabla g|_{\mathbf{P}}$ el gradiente y `H` el Hessiano (matriz de segundas derivadas parciales) de g particularizados en $\mathbf{P}$. $(\nabla g)_i = \left.\dfrac{\partial g}{\partial x_i}\right|_{\mathbf{P}}$, $[\mathtt{H}]_{i,j} = \left.\dfrac{\partial^2 g}{\partial x_i \partial x_j}\right|_{\mathbf{P}}$

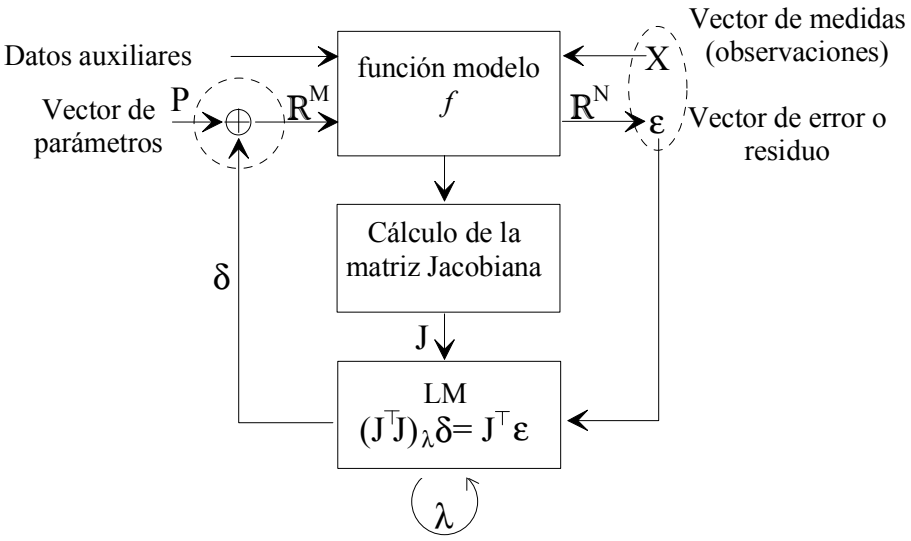


Figura D.1: Algoritmo de optimización Levenberg-Marquardt (LM)

Método de Newton

En el método de Newton se elige δ de tal forma que el gradiente en el siguiente punto de iteración, $\mathbf{P}+\delta$, sea cero (condición de extremo relativo). Tomamos el gradiente de la expresión D.1

$$\nabla g|_{\mathbf{P}+\delta} = \nabla g|_{\mathbf{P}} + \mathtt{H}\delta = 0$$

El sistema en forma estándar ($\mathtt{A}\mathbf{x} = \mathbf{b}$) es:

$$\mathtt{H}\delta = -\nabla g|_{\mathbf{P}} \tag{D.2}$$

Cuya solución es

$$\delta = \mathtt{H}^{-1}(-\nabla g|_{\mathbf{P}}) \tag{D.3}$$

En ésta última expresión se aprecia que el paso δ es el producto de dos términos: el inverso del Hessiano y el vector opuesto al gradiente, es decir, la dirección de máximo decrecimiento de la función g.

Método de Levenberg-Marquardt

Si la aproximación D.1 es buena, con el método de Newton sabemos cómo llegar al mínimo en un sólo paso (obtiene el paso δ que lleva de los parámetros actuales a los minimizantes).

$$\mathbf{P}_{\text{mín}} = \mathbf{P}_{\text{actual}} + \mathtt{H}^{-1}(-\nabla g|_{\mathbf{P}_{actual}}) \tag{D.4}$$

En cambio, si D.1 es una mala aproximación a la forma de la función g en $\mathbf{P}_{\text{actual}}$, lo mejor es explorar en la dirección opuesta al gradiente, como método de máxima pendiente ("Steepest descent method").

$$\mathbf{P}_{\text{siguiente}} = \mathbf{P}_{\text{actual}} + \text{cte}\,(-\nabla g|_{\mathbf{P}_{actual}}) \tag{D.5}$$

donde la constante es suficientemente pequeña para no agotar la dirección cuesta abajo.

La modificación que Levenberg-Marquardt hacen al método de Newton es, dada la semejanza de las ecuaciones que definen los pasos en ambos casos, unificarlas en una sola, con un parámetro λ. En la ecuación D.2, se multiplica la diagonal principal de $\mathtt{H}$ por $(1+\lambda)$.

$$\begin{array}{lcll} \mathtt{H}^*_{ii} & = & \mathtt{H}_{ii}(1+\lambda) & \text{diagonal principal} \\ \mathtt{H}^*_{ij} & = & \mathtt{H}_{ij} & \text{si } i \neq j \end{array}$$

Y las ecuaciones (D.4) y (D.5) quedan:

$$\mathbf{P}_{\text{mín}} = \mathbf{P}_{\text{actual}} + (\mathtt{H}^*)^{-1}(-\nabla g|_{\mathbf{P}_{actual}}) \tag{D.6}$$

Así, cuando $\lambda \to 0$, el método LM es el de Newton (también llamado "Método del inverso del Hessiano") y cuando λ es muy grande, la matriz Hessiana aumentada $\mathtt{H}^*$ es diagonalmente dominante y el método LM se convierte en el método de máxima pendiente. La constante λ sirve para variar suavemente entre estos dos extremos, utilizándose el último método lejos del mínimo (fase de exploración) y cambiando al de Newton a medida que se acerca al mínimo (fase de explotación). Una justificación similar se puede encontrar en [1, pág. 570].

DADA una estimación inicial de los parámetros $\mathbf{P}$ y una función (de coste) a minimizar: $g : \mathbf{P} \to \mathbb{R}$.

OBJETIVO Encontrar el vector de parámetros $\mathbf{P}$ que minimiza la función g.

ALGORITMO

(i) Inicializar una constante $\lambda = 0{,}001$ (valor típico).
(ii) Calcular $g(\mathbf{P})$, el gradiente y el Hessiano de la función en $\mathbf{P}$.
(iii) Crear la ecuación a resolver: $\mathtt{H}\delta = -\nabla g|_{\mathbf{P}}$, es decir, $\mathtt{N}\delta = \varepsilon$
(iv) Aumentar la ecuación: $\mathtt{H}^*\delta = -\nabla g|_{\mathbf{P}}$
(v) Resolver $\delta = (\mathtt{H}^*)^{-1}(-\nabla g|_{\mathbf{P}})$
(vi) Calcular el nuevo valor $g(\mathbf{P} + \delta)$.
(vii) Si $g(\mathbf{P} + \delta) < g(\mathbf{P})$, se actualiza el vector de parámetros $\mathbf{P} = \mathbf{P} + \delta$, se disminuye λ en un factor de 10 y se empieza de nuevo en (ii) o se termina.
(viii) Si $g(\mathbf{P} + \delta) \geq g(\mathbf{P})$, se incrementa λ en un factor de 10 y se vuelve a (iv) (se conserva $\mathbf{P}$).

Programación del algoritmo en `LMBchi2`:

ENTRADA:

`P0`	estimación inicial del vector de parámetros $\mathbf{P}$
`fcoste`	nombre de la función de coste $g : \mathbf{P} \mapsto \mathbb{R}$
`varargin`	argumentos opcionales que necesite la función `fcoste`. `fcoste(P,varargin:);`

SALIDA:

`P0`	vector de parámetros del mínimo
`coste_min`	valor del mínimo (de la función de coste)

Observaciones

El sufijo "`chi2`" del nombre de la rutina lo puse para recordar que es el algoritmo LM programado al que hay que pasarle la función a minimizar, no el modelo. En [31], capítulo 15, simbolizan dicha función como χ^2, la *función de mérito*.

Sin duda alguna, el paso más costoso es el cálculo del Hessiano de la función (aproximación numérica). Se pueden utilizar como subrutinas para esta tarea `hessiano` o `hessianofd`. En general, no ofrece resultados tan buenos como los proporcionados por las rutinas que minimizan explotando el conocimiento de una función modelo. En tales rutinas, se utiliza la matriz jacobiana, que requiere menos evaluaciones de la función para estimarla numéricamente. Además, si la rutina va a ser aplicada a un problema concreto del que se conoce exactamente el modelo, el cálculo del Hessiano es menos costoso porque se puede obtener analíticamente y de forma exacta, sin aproximaciones por diferencias finitas.

D.1.3. Cuando el vector de medidas objetivo es el nulo

La pregunta que ha inspirado este apartado es: ¿Qué sucede en el desarrollo si el vector de medidas objetivo que se desea alcanzar es $\mathbf{X} = \mathbf{0}$?

En este apartado se pretende relacionar el problema lineal

$$\min_{\mathbf{x}} \|\mathbf{b} - \mathtt{A}\mathbf{x}\|^2$$

con el problema no lineal

$$\min_{\mathbf{P}} \|\mathbf{X} - f(\mathbf{P})\|^2$$

Este último se reduce al primero si $\mathbf{P} = \mathbf{x}$, $\mathbf{X} = \mathbf{b}$ y la función f es lineal $f(\mathbf{P}) = \mathtt{A}\mathbf{P}$.

Hasta ahora se ha visto la solución del algoritmo LM al problema en el que tenemos una función modelo no lineal:

$$\begin{array}{rccc} f: & \mathbb{R}^M & \longrightarrow & \mathbb{R}^N \\ & \mathbf{P} & \mapsto & \widehat{\mathbf{X}} \end{array}$$

Y buscamos el vector $\mathbf{P} \in \mathbb{R}^M$, tal que $\mathbf{X} = f(\mathbf{P}) + \epsilon$, con mínimo error $\|\epsilon\|^2$. Es equivalente a

$$\min_{\mathbf{P}} \|\mathbf{X} - f(\mathbf{P})\|^2 = \min \|\epsilon\|^2$$

Los pasos del algoritmo son:

1. Elegir un punto inicial $\mathbf{P}_0$
2. Calcular el error en la aproximación: $\epsilon_0 = \mathbf{X} - f(\mathbf{P}_0)$
3. Suponer un que la función modelo es localmente lineal, para obtener una expresión de la variación de la salida cuando varía la entrada: $f(\mathbf{P}_0 + \Delta) = f(\mathbf{P}_0) + \mathtt{J}\Delta$, siendo $\mathtt{J} = \partial f(\mathbf{P})/\partial \mathbf{P}$ la matriz jacobiana de la función, evaluada en $\mathbf{P}_0$.
4. Buscamos un vector de parámetros $\mathbf{P}_1 = \mathbf{P}_0 + \Delta$ que minimiza

$$\|\mathbf{X} - f(\mathbf{P}_1)\|^2 = \|\mathbf{X} - f(\mathbf{P}_0) - \mathtt{J}\Delta\|^2 = \|\epsilon_0 - \mathtt{J}\Delta\|^2$$

5. Esta última expresión tiene solución lineal porque hemos supuesto que la función modelo es localmente lineal, por dos métodos directos: las ecuaciones normales y la pseudoinversa .
   a) Ecuaciones Normales: $\mathtt{J}^\top \mathtt{J}\Delta = \mathtt{J}^\top \epsilon_0$
   b) Pseudoinversa: $\Delta = \mathtt{J}^+ \epsilon_0$

   El algoritmo de LM modifica las ecuaciones normales, así que no utiliza la pseudoinversa. Ecuaciones Normales modificadas (ampliadas): $(\mathtt{J}^\top \mathtt{J})_\lambda \Delta = \mathtt{J}^\top \epsilon_0$
6. Se actualiza el vector de parámetros y se continua iterando, en función del nuevo coste obtenido, según el parámetro λ.

Pensemos otra versión del algoritmo LM, en la que la función modelo es una aplicación del vector de parámetros $\mathbf{P}$ al vector de error o residuo ϵ. Esto es equivalente a la formulación anterior si el vector de medidas es nulo $\mathbf{X} = \mathbf{0}$.

$$\begin{array}{rccc} f: & \mathbb{R}^M & \longrightarrow & \mathbb{R}^N \\ & \mathbf{P} & \mapsto & \epsilon \end{array}$$

Y buscamos el vector $\mathbf{P} \in \mathbb{R}^M$, tal que $\mathbf{0} = f(\mathbf{P}) + \epsilon$, con mínimo error $\|\epsilon\|^2$. Es decir, tal que $\epsilon = -f(\mathbf{P})$, con mínimo error $\|\epsilon\|^2$. Esto equivale a

$$\min_{\mathbf{P}} \|\mathbf{0} - f(\mathbf{P})\|^2 = \min_{\mathbf{P}} \| - f(\mathbf{P})\|^2 = \min \|\epsilon\|^2$$

Veamos cómo se modifican los pasos del algoritmo:

1. Elegir un punto inicial $\mathbf{P}_0$

2. Calcular el error en la aproximación: $\epsilon_0 = -f(\mathbf{P}_0)$

3. Suponer un que la función modelo es localmente lineal, para obtener una expresión de la variación de la salida cuando varía la entrada: $f(\mathbf{P}_0 + \Delta) = f(\mathbf{P}_0) + \mathtt{J}\Delta$, siendo $\mathtt{J} = \partial f(\mathbf{P})/\partial \mathbf{P}$ la matriz jacobiana de la función, evaluada en $\mathbf{P}_0$. La matriz J no depende del vector de medidas objetivo (anterior $\mathbf{X}$).

4. Buscamos un vector de parámetros $\mathbf{P}_1 = \mathbf{P}_0 + \Delta$ que minimiza

$$\| - f(\mathbf{P}_1)\|^2 = \| - f(\mathbf{P}_0) - \mathtt{J}\Delta\|^2 = \|\epsilon_0 - \mathtt{J}\Delta\|^2$$

5. Esta última expresión tiene solución lineal porque hemos supuesto que la función modelo es localmente lineal. Las Ecuaciones Normales modificadas (ampliadas) son: $(\mathtt{J}^\top \mathtt{J})_\lambda \Delta = \mathtt{J}^\top \epsilon_0$

6. Se actualiza el vector de parámetros y se continua iterando, en función del nuevo coste obtenido, según el parámetro λ.

Esta segunda formulación es la que emplea la rutina `lsqnonlin`: se le pasa el nombre de una función que devuelve un residuo ϵ dado un vector de parámetros . Así que ya sabemos programar una rutina como `lsqnonlin`: basta emplear alguna de las LM que tenemos, cambiando la función modelo para que el vector de medidas objetivo sea el vector nulo. Esta versión es la que se utiliza en la minimización iterativa de la distancia algebraica en la matriz fundamental, el tensor trifocal o el tensor cuadrifocal.

D.1.4. Cálculo de la matriz jacobiana

`fdjac`

Es una rutina adaptada de [31], sección 9.7 e implementa una diferenciación numérica mediante el método de las diferencias finitas (hacia delante). Dada una función $f : \mathbf{x} \in \mathbb{R}^M \to \mathbb{R}^N$, la matriz jacobiana es

$$\mathtt{J}(f) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_M} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_M} \end{bmatrix}$$

Tiene dimensiones $M \times N$: tantas columnas como variables independientes (en el vector $\mathbf{x}$) y tantas filas como variables devuelva la función.

En MATLAB, se puede calcular esta matriz por columnas con un único bucle `for`. En cada pasada se incrementa una variable independiente, x_i, en una cantidad h, se evalúa la función vectorial $f(\tilde{\mathbf{x}}_i) = f(x_1, x_2, \ldots, x_i + h, \ldots, x_m)$ para esa entrada y se obtiene la columna i-ésima de J como $(f(\tilde{\mathbf{x}}_i) - f(\mathbf{x}))/h$. Buenos valores de h están entre 10^{-8} y 10^{-4}.

En resumen, se puede construir una aproximación a la matriz jacobiana con $M + 1$ evaluaciones de la función (vectorial). En nuestro contexto, esta función vectorial es la función modelo, $f : \mathbf{P} \in \mathbb{R}^M \to \widehat{\mathbf{X}} \in \mathbb{R}^N$.

`fdjac_selectivo`

Calcula la matriz jacobiana mediante la aproximación por diferencias finitas hacia delante, pero sólo las columnas indicadas en su llamada. Aplicación: si tenemos un vector de parámetros y sólo se desea optimizar respecto a ciertos elementos (queremos dejar fijos varios parámetros), esta rutina calcula la matriz jacobiana respecto de los parámetros que varían, evitando cálculos innecesarios de las columnas

de ceros debidas a los parámetros fijos. Esto se utiliza en la rutina `LMB` (§ D.1.1).

`cdjac`

Además, se ha profundizado en el tema y como resultado han sido implementadas otras dos rutinas, que realizan la misma función de aproximación de la matriz jacobiana que `fdjac`, pero mediante diferencias finitas centrales (extrapolación de Richardson): `cdjac` y `cdjac_5point` , por lo que el error de aproximación es menor a costa de realizar más evaluaciones.

La rutina `cdjac` utiliza la fórmula de los 3 puntos en diferencias centrales:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \text{error}$$

$$\text{error} = f^{(3)}(\xi)\frac{h^2}{6} = \mathcal{O}(h^2), \quad \text{con} \;\; \xi \in [x-h, x+h]$$

`cdjac_5point`

Aproxima la derivada por la fórmula de los 5 puntos en diferencias centrales:

$$f'(x) = \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h} + \text{error}$$

$$\text{error} = f^{(5)}(\xi)\frac{h^4}{30} = \mathcal{O}(h^4), \quad \text{con} \;\; \xi \in [x-h, x+h]$$

D.1.5. Cálculo del gradiente y del Hessiano

El gradiente es la matriz jacobiana de una función de $\mathbb{R}^M \to \mathbb{R}$, es decir, como una fila de dicha matriz (aunque según el convenio de utilizar vectores columna, hay que transponer dicha fila). Así, para calcular el gradiente se utiliza la misma rutina que para calcular la matriz jacobiana, `fdjac`.

$$\nabla g = \left[\begin{array}{cccc} \frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2} & \cdots & \frac{\partial g}{\partial x_m} \end{array} \right]^\top$$

El Hessiano es la matriz de segundas derivadas parciales de una función $g : \mathbf{x} \in \mathbb{R}^M \to \mathbb{R}$, tiene dimensiones $M \times M$.

$$\text{Hessiano}(g) = \left[\begin{array}{cccc} \frac{\partial^2 g}{\partial x_1^2} & \frac{\partial^2 g}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 g}{\partial x_1 \partial x_M} \\ \frac{\partial^2 g}{\partial x_2 \partial x_1} & \frac{\partial^2 g}{\partial x_2^2} & \cdots & \frac{\partial^2 g}{\partial x_2 \partial x_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 g}{\partial x_M \partial x_1} & \frac{\partial^2 g}{\partial x_M \partial x_2} & \cdots & \frac{\partial^2 g}{\partial x_M^2} \end{array} \right] \equiv \left[\begin{array}{cccc} g_{x_1 x_1} & g_{x_1 x_2} & \cdots & g_{x_1 x_M} \\ g_{x_2 x_1} & g_{x_2 x_2} & \cdots & g_{x_2 x_M} \\ \vdots & \vdots & \ddots & \vdots \\ g_{x_M x_1} & g_{x_M x_2} & \cdots & g_{x_M x_M} \end{array} \right]$$

Para calcular el Hessiano estamos suponiendo que la función g es de clase $\mathcal{C}^2$ (con primera y segunda derivada continua). Basta que g sea de clase $\mathcal{C}^1$ para que el Hessiano sea simétrico, por lo que sólo hace falta calcular $M(M+1)/2$ derivadas segundas (la mitad de los elementos, aproximadamente).

Se proponen dos rutinas para calcular dicho Hessiano:

`hessianofd`

Es la más ingeniosa de las dos y se basa en la observación de que el Hessiano es la matriz jacobiana del gradiente de la función. Hessiano$(g) =$ `J`(∇g). La programación se hace en una línea: llama a la rutina de cálculo de la matriz jacobiana (`fdjac`) con dos niveles de anidamiento: el primero calcula el gradiente de la función y el segundo calcula la matriz jacobiana del gradiente.

Utiliza diferencias finitas hacia delante y no explota la simetría del Hessiano. En cambio, se puede decir que es la "rápida" de las dos programadas, ya que necesita $(M+1)^2$ evaluaciones de la función para aproximar el Hessiano. Explicación: para obtener el gradiente, se necesitan $(M+1)$ evaluaciones de la función y para calcular la matriz jacobiana del gradiente, se realizan $(M+1)$ evaluaciones del gradiente. Como no se explota la simetría del Hessiano, se realizan aproximadamente el doble de evaluaciones de las necesarias: $(M+1)^2 \approx 2\,(M(M+1)/2)$.

`hessiano`

Obtiene los elementos del Hessiano de despejar en el desarrollo en serie de Taylor de la función g. Supongamos una función de dos variables: $g(x_i, x_j)$, dicho desarrollo en serie es:

$$\begin{aligned} g(x_i+\Delta x_i, x_j+\Delta x_j) &= g(x_i,x_j) + [g_{x_i}(x_i,x_j)\Delta x_i + g_{x_j}(x_i,x_j)\Delta x_j] \\ &+ \frac{1}{2!}[g_{x_ix_i}(x_i,x_j)(\Delta x_i)^2 + 2g_{x_ix_j}(x_i,x_j)\Delta x_i\Delta x_j + g_{x_jx_j}(x_i,x_j)(\Delta x_j)^2] \\ &+ \frac{1}{3!}[g_{x_ix_ix_i}(x_i,x_j)(\Delta x_i)^3 + 3g_{x_ix_ix_j}(x_i,x_j)(\Delta x_i)^2\Delta x_j \\ &+ 3g_{x_ix_jx_j}(x_i,x_j)\Delta x_i(\Delta x_j)^2 + g_{x_jx_jx_j}(x_i,x_j)(\Delta x_j)^3] + \ldots \end{aligned}$$

Para los elementos de la diagonal principal hacen falta 2 evaluaciones de la función además de la evaluación en el punto de particularización del Hessiano y el error residual es aproximadamente de $\mathcal{O}(h^3)$ $(h = \text{máx}\,\{\Delta x, \Delta y\})$ porque se cancelan todas las terceras derivadas parciales.

$$g_{x_ix_i}(x_ix_j) = \frac{g(x_i+\Delta x_i, x_j) + g(x_i-\Delta x_i, x_j) - 2g(x_i,x_j)}{(\Delta x_i)^2}$$

Para cada elemento no situado en la diagonal principal se requieren 4 evaluaciones de la función y el error residual es también de $\mathcal{O}(h^3)$.

$$g_{x_ix_j}(x_ix_j) = \frac{g(x_i+\Delta x_i, x_j+\Delta x_j) + g(x_i-\Delta x_i, x_j-\Delta x_j) - g(x_i+\Delta x_i, x_j-\Delta x_j) - g(x_i-\Delta x_i, x_j+\Delta x_j)}{4\Delta x_i\Delta x_j}$$

Las fórmulas también son válidas en el caso de una función de más de dos variables, ya que para calcular el elemento Hessiano$(g)_{i,j}$ no influyen más variables que x_i y x_j. En total hacen falta, explotando la propiedad de simetría de la matriz, $(2M+1) + 4(M^2-M)/2 = 2M^2+1$. Utiliza diferencias finitas centrales. Para el mismo valor de h, la aproximación es mejor (error residual de orden superior: $\mathcal{O}(h^3)$ en lugar de $\mathcal{O}(h^2)$), aunque tarda casi el doble debido al número de evaluaciones de la función: $2M^2+1 \approx 2(M+1)^2$. Para pasar de un error $\mathcal{O}(h^2)$ a uno $\mathcal{O}(h^3)$ se cuadruplica el número mínimo de evaluaciones necesarias, aproximadamente.

D.1.6. Algoritmo particionado

Nos referimos al algoritmo de Levenberg-Marquardt Particionado Básico, algoritmo A4.1 de [1, pág. 572-575]. Está bien descrito en esas páginas, las cuales son una lectura recomendada si se quiere entender el código de la rutina `LMPB`, que es la que lo implementa.

Básicamente, se decide del enunciado del problema que el vector de parámetros $\mathbf{P}$ se puede dividir en dos partes $\mathbf{P} = (\mathbf{a}^\top, \mathbf{b}^\top)^\top$ y se reformula el algoritmo `LMB` de tal forma que incluya las modificaciones oportunas para sacar provecho de la situación: las matrices a invertir y los sistemas a resolver son de menor dimensión (menos costosos computacionalmente y rápidos).

Los argumentos de entrada y salida son los mismos que los de la rutina `LMB`, salvo que se debe proporcionar un argumento de entrada más respecto a los de aquella y es `na`, el número de variables del vector $\mathbf{a}$ dentro de $\mathbf{P}$. Las mismas consideraciones son aplicables al cálculo de la matriz de covarianzas $\Sigma_{\mathbf{P}}$.

Internamente, llama la rutina `fdjac` para calcular la matriz jacobiana numéricamente y luego la divide en dos partes $\mathtt{J} = [\mathtt{A} \,|\, \mathtt{B}]$ según el valor de `na`.

Aplicación: se puede utilizar para estimar homografías entre imágenes, la matriz fundamental, el ajuste de haces (Bundle Adjustment) en la calibración proyectiva, etc. En estas situaciones, $\mathbf{a}$ suele definir la transformación y $\mathbf{b}$ las correcciones de los puntos del ajuste (en la primera imagen o en el espacio).

D.1.7. Algoritmo particionado y disperso

La rutina `LMPSI` es la implementación del algoritmo de Levenberg-Marquardt Particionado Disperso (Sparse) con $\Sigma_{\mathbf{X}} = \mathrm{Id}$, algoritmo A4.3 de [1, pág. 575-577].

Está pensado para problemas en los que el vector de parámetros $\mathbf{P}$ se puede dividir de la forma $\mathbf{P} = (\mathbf{a}^\top, \mathbf{b}_1^\top, \ldots, \mathbf{b}_n^\top)^\top$, el vector de medidas $\mathbf{X} = (\mathbf{X}_1^\top, \ldots, \mathbf{X}_n^\top)^\top$ y se verifica la condición de dispersión (yo la llamaría de independencia), en la que cada $\widehat{\mathbf{X}}_i$ sólo depende de $\mathbf{a}$ y $\mathbf{b}_i$, NO depende de otros $\mathbf{b}_j$, es decir, $\partial\widehat{\mathbf{X}}_i/\partial\mathbf{b}_j = 0$ para $i \neq j$.

Debido a esta división, la matriz jacobiana $\mathtt{J} = [\partial\widehat{\mathbf{X}}_i/\partial\mathbf{P}] = [\mathtt{A} \,|\, \mathtt{B}]$ es una matriz dispersa a bloques (`B` es diagonal a bloques) y se cambia el algoritmo `LMPB` para aprovechar esa estructura, tanto de `J` como de $\mathtt{J}^\top\Sigma_{\mathbf{X}}^{-1}\mathtt{J}$ si también se asume una estructura dispersa particular para $\Sigma_{\mathbf{X}}$. Todo ello hace que en cada paso del algoritmo el tiempo de cómputo sea lineal con n. Sin la condición de dispersión, la complejidad es de orden n^3.

Citemos la entrada-salida:

Entrada:

`X`	vector de medidas u observaciones objetivo
`P0`	estimación inicial del vector de parámetros $\mathbf{P}$
`func`	nombre de la función modelo $f : \mathbf{P} \mapsto \widehat{\mathbf{X}}$
`na`	número de variables del vector $\mathbf{a}$ dentro de $\mathbf{P}$
`tam_bi`	tamaño de cada componente $\mathbf{b}_i$ del vector de parámetros
`tam_Xi`	tamaño de cada componente $\widehat{\mathbf{X}}_i$ del vector de medidas

Salida:

`P0`	vector de parámetros del mínimo
`coste_min`	valor del mínimo (de la función de coste)
`X_min`	valor del vector de medidas en el mínimo
`Ep`	(opcional) matriz de covarianzas de los parámetros estimados (`P0`)

Como se aprecia, se deben proporcionar tres argumentos de entrada más respecto a los de `LMB`: `na`, `tam_bi` y `tam_Xi`.

Internamente, llama la rutina `fdjac` para calcular la matriz jacobiana. A continuación selecciona las submatrices de ésta donde no es seguro que existan ceros debido a la condición de dispersión.

Las rutinas `LMPSHI_afin` y `LMPSFI` son particularizaciones de esta rutina y lo único que las distingue es la construcción de la matriz jacobiana: en éstas últimas se construyen de forma exacta a partir de los datos ya que se utilizan en una aplicación concreta, no son de propósito general. Las derivadas exactas son más rápidas y precisas para aplicaciones concretas.

La subrutina de cálculo de la matriz de covarianzas $\Sigma_{\mathbf{P}}$ es un poco diabólica para sacar partido de la estructura dispersa a bloques de $\mathtt{J}^\top\mathtt{J}$ (hemos supuesto $\Sigma_{\mathbf{X}} = \mathrm{Id}$).

D.2. Manipulación de matrices

Matrices de 3 dimensiones

La función `matriz3dim2matriz2dim` convierte una matriz `M` de 3 dimensiones $n1 \times n2 \times n3$ en una matriz de 2 dimensiones, concatenando las matrices de la tercera dimensión vertical u horizontalmente, según se indique. Devuelve un error si el tipo de concatenación no es `'h'` ni `'v'`.

`Y = matriz3dim2matriz2dim(M,'h')` produce la matriz $[\mathtt{M}(:,:,1), \mathtt{M}(:,:,2), \ldots, \mathtt{M}(:,:,n3)]$

`Y = matriz3dim2matriz2dim(M,'v')` produce $\begin{bmatrix} \mathtt{M}(:,:,1) \\ \mathtt{M}(:,:,2) \\ \vdots \\ \mathtt{M}(:,:,n3) \end{bmatrix}$

Se utiliza en `LMPSFI` y `LMPSHI_afin` para construir la parte `A` de la matriz jacobiana en la subrutina local del cálculo de la matriz de covarianzas.

La función que se puede utilizar como rutina inversa a la anterior es `matriz2dim2matriz3dim`, la cual convierte una matriz de 2 dimensiones `M` en una matriz de 3 dimensiones, verticalmente en bloques de bloques de $n \times$ `size(y,2)` u horizontalmente en bloques de `size(y,1)` $\times\, n$ elementos. También devuelve un error si el tipo de escisión no es `'h'` ni `'v'`.

Matrices diagonales a bloques

La función `matriz3dim2blkdiag` convierte una matriz `Bi` de 3 dimensiones $n1 \times n2 \times n3$ en una matriz 2D diagonal por bloques.

`B = matriz3d2blkdiag(Bi)` produce $\begin{bmatrix} \mathtt{Bi}(:,:,1) & 0 & \cdots & 0 \\ 0 & \mathtt{Bi}(:,:,2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathtt{Bi}(:,:,n3) \end{bmatrix}$

También se proporciona la rutina inversa: `blkdiag2matriz3dim`, que convierte una matriz diagonal por bloques `B` de $n1 \times n2$ en un array `Bi` de 3 dimensiones $n1 \times n2 \times n3$. Devuelve un error si las dimensiones $n1$ y $n2$ no son el mismo divisor de las correspondientes dimensiones de `B`.

Se utilizan en `LMPSI`, `LMPSFI` y `LMPSHI_afin` para componer y descomponer información de las matrices `V` y `B`, que son diagonales a bloques.

Matrices a bloques

`blockmatrix2matrix` convierte una matriz de 4 dimensiones `Mij` en una matriz de 2 dimensiones `M` con la siguiente estructura:

$$\mathtt{M} = \begin{bmatrix} \mathtt{Mij}(:,:,1,1) & \mathtt{Mij}(:,:,1,2) & \cdots & \mathtt{Mij}(:,:,1,\mathtt{nbc}) \\ \mathtt{Mij}(:,:,2,1) & \mathtt{Mij}(:,:,2,2) & \cdots & \mathtt{Mij}(:,:,2,\mathtt{nbc}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathtt{Mij}(:,:,\mathtt{nbf},1) & \mathtt{Mij}(:,:,\mathtt{nbf},2) & \cdots & \mathtt{Mij}(:,:,\mathtt{nbf},\mathtt{nbc}) \end{bmatrix}$$

Su rutina inversa, `matrix2blockmatrix`, convierte una matriz de 2 dimensiones `M` en una matriz de 4 dimensiones `Mij` en bloques de tamaño indicado: `nf` $\times$ `nc`, por filas y columnas.

D.3. Afinidades

El punto de partida en el esquema global de un sistema completo de autocalibración y reconstrucción 3D es una secuencia de imágenes. El convenio que se utiliza en el tratamiento digital de imágenes es poner el sistema de coordenadas en la esquina superior izquierda de la imagen, con el eje horizontal hacia la derecha como eje x positivo, y el eje vertical hacia abajo como eje y positivo. Dadas las imágenes de la secuencia, obtenemos los puntos característicos (esquinas), los cuales han sido detectados y seguidos mediante el algoritmo de seguimiento (*tracking*). Estos puntos están referidos al sistema de coordenadas de la imagen. Si las imágenes son de 576 filas y 720 columnas, las coordenadas de los puntos estarán en los intervalos: $0 \leq x \leq 720, 0 \leq y \leq 576$. Las coordenadas homogéneas de un punto de la imagen serán, en media $(360, 288, 1)^\top$. Así pues, tenemos todos los puntos localizados en el primer cuadrante de $\mathbb{R}^2$ y existe una gran diferencia entre los órdenes de magnitud de las dos primeras coordenadas y la tercera. Además existe un *offset* intrínseco debido a que los puntos no están centrados en el sistema de referencia: en términos eléctricos diríamos que hay una componente continua (el centroide de los puntos en el plano).

Para mejorar la estabilidad numérica - condicionamiento de los sistemas de ecuaciones, se realiza una normalización afín de las coordenadas de los puntos en las imágenes, que consiste en una traslación más un escalado: se trasladan los puntos para que su centroide sea el origen de coordenadas (así se elimina la "componente continua" del resto de los cálculos), y a continuación se escalan los puntos para que su distancia media al origen sea la raíz cuadrada de la dimensión del espacio al que pertenecen: $\mathbb{P}^n \Rightarrow \sqrt{n}$. La razón de estos dos criterios es conseguir que las coordenadas homogéneas de los puntos resultantes estén equilibradas. Para los puntos en las imágenes, los puntos normalizados resultantes son del orden del $(1, 1, 1)^\top$, como corresponde a puntos de $\mathbb{P}^2$.

Este cambio lo hizo popular R. Hartley en su algoritmo normalizado de los 8 puntos para el cálculo de la matriz fundamental. También está citado en su libro [1, pág. 92]. No sólo se aplica allí, sino en la mayor parte de los algoritmos posteriores de otros autores.

A continuación, se procede a ver cómo afecta este cambio al resto de los objetos, en especial a los que sirven para calibrar: proyección de la cónica absoluta, su dual, cuádrica absoluta dual, plano del infinito, etc.

Sea el modelo de proyección lineal: $\mathbf{x} \sim \mathtt{P}\mathbf{X}$. Si se realiza un cambio de coordenadas del tipo $\mathbf{x}_n = \mathtt{T}\mathbf{x}$ sobre los puntos proyectados, ¿Cómo actúa en el resto de los elementos? Para responder a esta pregunta se utilizarán relaciones ya desarrolladas en el capítulo 3.

Para preservar la relación de incidencia entre puntos y rectas de $\mathbb{P}^2$, $\mathbf{x}^\top\mathbf{l} = \mathbf{l}^\top\mathbf{x} = 0$ es inmediato comprobar que las rectas se transforman de la forma $\mathbf{l}_n = \mathtt{T}^{-\top}\mathbf{l}$, ya que $\mathbf{x}^\top\mathbf{l} = (\mathtt{T}^{-1}\mathbf{x}_n)^\top\mathbf{l} = \mathbf{x}_n^\top\mathtt{T}^{-\top}\mathbf{l} =$

$\mathbf{x}_n^\top \mathbf{l}_n \Rightarrow \mathtt{T}^{-\top}\mathbf{l} = \mathbf{l}_n$.

Insertando el cambio de coordenadas en la ecuación de la proyección $\mathbf{x} \sim \mathtt{P}\mathbf{X}$, concluimos que sólo cambia la matriz de proyección, no las coordenadas de los puntos 3D (de $\mathbb{P}^3$) que se proyectan sobre los puntos en las imágenes. $\mathbf{x}_n = \mathtt{T}\mathbf{x} \sim \mathtt{TP}\mathbf{X} \sim \mathtt{P}_n\mathbf{X}$, siendo $\mathtt{P}_n = \mathtt{TP}$.

Suponiendo que se trabaja en un sistema de referencia métrico (o euclídeo) de $\mathbb{P}^3$, la matriz de proyección admite una descomposición: $\mathtt{P} = \mathtt{KR}[\mathtt{I} \mid -\widetilde{\mathbf{C}}]$. Por lo que la matriz de proyección tras el cambio de coordenadas es $\mathtt{P}_n = \mathtt{TKR}[\mathtt{I} \mid -\widetilde{\mathbf{C}}] = \mathtt{K}_n\mathtt{R}[\mathtt{I} \mid -\widetilde{\mathbf{C}}]$, siendo $\mathtt{K}_n = \mathtt{TK}$ la matriz de parámetros intrínsecos tras el cambio.

Estamos en condiciones de ver cómo se transforma la DIAC, dual de la proyección de la cónica absoluta. La DIAC respecto de los datos originales es $\omega^* = \mathtt{K}\mathtt{K}^\top$. Y tras el cambio: $\omega_n^* = \mathtt{K}_n\mathtt{K}_n^\top = \mathtt{TK}(\mathtt{TK})^\top = \mathtt{TKK}^\top\mathtt{T}^\top = \mathtt{T}\omega^*\mathtt{T}^\top$.

La dual de la DIAC, la IAC o proyección de la cónica absoluta es $\omega = (\omega^*)^{-1}$. Una vez realizado el cambio, $\omega_n = (\omega_n^*)^{-1} = (\mathtt{T}\omega^*\mathtt{T}^\top)^{-1} = \mathtt{T}^{-\top}(\omega^*)^{-1}\mathtt{T}^{-1} = \mathtt{T}^{-\top}\omega\mathtt{T}^{-1}$.

Queda por ver cómo se transforman los elementos del espacio: la cuádrica absoluta dual, la cónica absoluta y el plano del infinito. Como el cambio de coordenadas del plano sólo influye en la matriz de proyección (multiplicando por la izquierda) y los puntos 3D no cambian, el resto de objetos del espacio tampoco sufren cambio alguno debido al cambio de coordenadas en el plano de la imagen. Comprobémoslo:

La DIAC se obtiene de la cuádrica absoluta dual y las matrices de proyección mediante la fórmula: $\omega^* = \mathtt{P}\mathtt{Q}_\infty^*\mathtt{P}^\top$. Para la DIAC tras el cambio, $\omega_n^* = \mathtt{P}_n\mathtt{Q}_{\infty\,n}^*\mathtt{P}_n^\top \Leftrightarrow \mathtt{T}\omega^*\mathtt{T}^\top = \mathtt{TP}\mathtt{Q}_{\infty\,n}^*(\mathtt{TP})^\top = \mathtt{TP}\mathtt{Q}_{\infty\,n}^*\mathtt{P}^\top\mathtt{T}^\top$ Multiplicando en ambos miembros a la izquierda por $\mathtt{T}^{-1}$ y a la derecha por $\mathtt{T}^{-\top}$, resulta: $\omega^* = \mathtt{P}\mathtt{Q}_{\infty\,n}^*\mathtt{P}^\top$ y comparando con la expresión al principio de este párrafo, concluimos que $\mathtt{Q}_{\infty\,n}^* = \mathtt{Q}_\infty^*$. La cuádrica absoluta dual queda invariante a transformaciones en los planos de las imágenes.

Como la cuádrica absoluta dual no cambia, tampoco lo hace su núcleo, el plano del infinito π_∞. Y la cónica absoluta Ω_∞ (dual de $\mathtt{Q}_\infty^*$) tampoco cambia, por la misma razón. Resumimos los resultados en siguiente tabla D.1.

Punto de la imagen	$\mathbf{x}_n = \mathtt{T}\mathbf{x}$	$\mathbf{x} = \mathtt{T}^{-1}\mathbf{x}_n$
Matriz de Proyección	$\mathtt{P}_n = \mathtt{TP}$	$\mathtt{P} = \mathtt{T}^{-1}\mathtt{P}_n$
Matriz de parámetros intrínsecos	$\mathtt{K}_n = \mathtt{TK}$	$\mathtt{K} = \mathtt{T}^{-1}\mathtt{K}_n$
DIAC	$\omega_n^* = \mathtt{T}\omega^*\mathtt{T}^\top$	$\omega^* = \mathtt{T}^{-1}\omega_n^*\mathtt{T}^{-\top}$
IAC	$\omega_n = \mathtt{T}^{-\top}\omega\mathtt{T}^{-1}$	$\omega = \mathtt{T}^\top\omega_n\mathtt{T}$
Punto 3D	$\mathbf{X}_n = \mathbf{X}$	$\mathbf{X} = \mathbf{X}_n$
Cuádrica absoluta dual	$\mathtt{Q}_{\infty\,n}^* = \mathtt{Q}_\infty^*$	$\mathtt{Q}_\infty^* = \mathtt{Q}_{\infty\,n}^*$
Plano del infinito	$\pi_{\infty\,n} = \pi_\infty$	$\pi_\infty = \pi_{\infty\,n}$
Cónica absoluta	$\Omega_{\infty\,n} = \Omega_\infty$	$\Omega_\infty = \Omega_{\infty\,n}$

Cuadro D.1: Efecto de una normalización afín de los puntos en las imágenes sobre el resto de elementos relacionados con la calibración

La normalización realizada tiene ventajas sobre la IAC y la DIAC, porque consigue que sus 3 autovalores sean del mismo orden de magnitud. Como es sabido, la IAC y la DIAC tienen matrices definidas (positivas o negativas). En los algoritmos que estimen estos elementos, a veces la solución puede ser una matriz próxima a ser singular. En estos casos, se pueden incluir términos de penalización dentro de la función de coste, de tal forma que se impongan las restricciones propias de estos objetos. Tiene más sentido penalizar cuando los datos están normalizados que cuando no, ya que en el primer caso

hay mayor estabilidad numérica.

El esquema de utilización de las relaciones de normalización es el siguiente: se normalizan los puntos observados en las imágenes, se realiza una calibración proyectiva seguida del algoritmo de autocalibración, se desnormalizan los elementos de acuerdo con la segunda columna de la tabla D.1, se procede a la reconstrucción 3D de los puntos en el marco de referencia métrico dado por la calibración y por último se visualiza la reconstrucción.

D.3.1. Normalización afín

`normaliza`

ENTRADA: una matriz `P`, cuyas columnas son coordenadas de puntos afines en $\mathbb{R}^n$.
REALIZA: una normalización afín de las coordenadas.

1. Se realiza una traslación de los puntos para que su centroide sea el origen de coordenadas.
2. Se escalan los puntos para que la distancia media desde el origen sea $\sqrt{n}$.

Sirve para puntos del plano afín si `P` tiene 2 filas, puntos del espacio afín si `P` tiene 3 columnas, etc.
SALIDA: las coordenadas afines normalizadas de los puntos $\mathtt{P}_n$ y la transformación afín `T` que transforma las coordenadas `P` en las coordenadas normalizadas $\mathtt{P}_n$.

`NormalizAfin`

ENTRADA: una matriz `P`, cuyas columnas son coordenadas homogéneas de puntos en $\mathbb{P}^n$.
REALIZA: una transformación afín sobre las coordenadas homogéneas. A partir de `P`, calcula sus coordenadas afines (mediante la rutina `proy2afin`) y llama a `normaliza` para normalizar las mismas. Después vuelve a pasar a coordenadas homogéneas.
SALIDA: las coordenadas homogéneas normalizadas y la afinidad.

D.3.2. Homogeneizar y deshomogeneizar las coordenadas

`proy2afin`

ENTRADA: una matriz `x`, cuyas columnas son coordenadas homogéneas de puntos en $\mathbb{P}^n$.
REALIZA: el paso de coordenadas homogéneas a coordenadas afines dividiendo por la ultima coordenada de cada punto. Matricialmente, es muy fácil de programar en MATLAB mediante `repmat` en lugar de un bucle `for`.
SALIDA: cada columna de la matriz `y` son las coordenadas afines de un punto.

La rutina `DeshomogeneizaCoords` del proyecto ADREP-3D realiza lo mismo, salvo que devuelve coordenadas afines dentro de las coordenadas homogéneas. (La tercera fila de la matriz devuelta es toda ella de unos).

`HomogeneizaCoords`

ENTRADA: una matriz `x`, cuyas columnas son coordenadas afines de puntos en $\mathbb{R}^n$.
REALIZA: el paso a coordenadas homogéneas añadiendo un 1 como última coordenada.
SALIDA: cada columna de la matriz `xh` son las coordenadas homogéneas de un punto de $\mathbb{P}^n$.

D.4. Homografías de $\mathbb{P}^2$

`HDLT_B`

Implementa el algoritmo lineal de § 4.2 sin normalización afín de los datos. También es el algoritmo 3.1 de [1, pág. 73]. Se le conoce como "The basic Direct Linear Transformation (DLT) for 2D homografies" y sirve para calcular transformaciones proyectivas en $\mathbb{P}^2$. Como se puede intuir, la `B` es de básico, para distinguir esta rutina de la que utiliza una normalización afín de los puntos (`HDLT_NA`).

Ofrece básicamente dos procedimientos de cálculo del vector solución $\mathbf{h}$: mediante la SVD de $\mathtt{A} = \mathtt{UDV}^\top$ o mediante los autovalores y autovectores de $\mathtt{A}^\top\mathtt{A} = \mathtt{VD}^2\mathtt{V}^{-1}$.

ENTRADA:

`P`	coordenadas homogéneas de los puntos de la primera imagen
`Q`	coordenadas homogéneas de los puntos de la segunda imagen
`tipoSol`	selecciona el tipo de optimización de la función de coste
`verb`	indica si se imprimen o no en la ventana de comandos el costes del ajuste

SALIDA:

`H`	matriz de la homografía
`coste`	mínima distancia algebraica

La variable `tipoSol` selecciona el tipo de cálculo empleado para hallar la matriz de la homografía estimada: si vale 1, mediante el comando `eig`, el cual calcula todos los autovalores y autovectores de $\mathtt{A}^\top\mathtt{A}$; si vale 2, mediante el comando `eigs`, que calcula los autovalores mayor y menor de $\mathtt{A}^\top\mathtt{A}$ y sus respectivos autovectores; si vale 0 y la opción por defecto, mediante la SVD de la matriz de diseño `A`.

Observación. Para construir la matriz de diseño `A` se puede realizar con un bucle `for` que recorra las parejas de puntos y utilice la siguiente observación: la matriz $\mathtt{A}_i$ se puede poner como el producto tensorial siguiente:

$$\mathtt{A}_i = \begin{bmatrix} \mathbf{0}^\top & -w_i'\mathbf{x}_i^\top & y_i'\mathbf{x}_i^\top \\ w_i'\mathbf{x}_i^\top & \mathbf{0}^\top & -x_i'\mathbf{x}_i^\top \\ -y_i'\mathbf{x}_i^\top & x_i'\mathbf{x}_i^\top & \mathbf{0}^\top \end{bmatrix} = \begin{bmatrix} 0 & -w_i' & y_i' \\ w_i' & 0 & -x_i' \\ -y_i' & x_i' & 0 \end{bmatrix} \otimes \mathbf{x}_i^\top = [\mathbf{x}_i']_\times \otimes \mathbf{x}_i^\top$$

El producto tensorial está implementado en MATLAB mediante el comando `kron` y el paso de vector a matriz antisimétrica es una rutina útil de programar, llamada `Cross2Matrix` en la implementación ADREP-3D.

HDLT_NA

Implementa el algoritmo lineal con normalización afín de los datos. También es el algoritmo 3.2 de [1, pág. 92], conocido como "The normalized Direct Linear Transformation (DLT) for 2D homografies". Las siglas `NA` significan "Normalización Afín". Posee la misma entrada-salida que `HDLT_B`.

Realiza una normalización afín (§ D.3) de las coordenadas de los puntos en ambas imágenes, después llama a `HDLT_B` y a continuación, desnormaliza la homografía solución.

HDLT_G

Es la variación del algoritmo lineal explicada en § 4.2. El interfaz y la salida son idénticos a los de `HDLT_B`. También tiene dos maneras de calcular la solución. El coste que devuelve se define de manera análoga a como se describió para `HDLT_B`, pero en lugar de valores singulares de `A`, se toman los valores singulares de `M`. También utiliza menos memoria que `HDLT_B`, ya que las matrices son más pequeñas. En definitiva, los algoritmos `HDLT_B` y `HDLT_G` se pueden utilizar en las mismas condiciones.

HDLT_NAG

Este algoritmo es la versión que emplea una normalización afín de los puntos previa a la llamada de la función `HDLT_G` y después deshace las transformaciones. Son aplicables las mismas comparaciones del apartado anterior, pero respecto de `HDLT_NA` y utiliza las mismas rutinas para realizar la normalización

que ésta.

homografia2D

Encapsula los algoritmos de cálculo de una homografía del plano comentados en el capítulo 4, sin utilizar RANSAC.

ENTRADA:

`x1`	coordenadas homogéneas de los puntos de la primera imagen
`x2`	coordenadas homogéneas de los puntos de la segunda imagen
`tipoFCoste`	selecciona la función de coste a minimizar
`tipoSol`	selecciona el tipo de optimización de la función de coste
`verb`	indica si se imprimen o no en la ventana de comandos el costes del ajuste

SALIDA:

`H`	matriz de la homografía que minimiza la función de coste elegida
`x1c`	puntos corregidos y normalizados (norma unidad) en la primera imagen en caso de elegir la función de coste del error de reproyección. En caso contrario se devuelven los puntos de entrada.
`x2c`	ídem para los puntos corregidos en la segunda imagen
`coste`	mínimo coste alcanzado
`Ep`	matriz de covarianzas $\Sigma_{\mathbf{p}}$ de los parámetros estimados, en el caso de utilizar el algoritmo de Levenberg-Marquardt.

Funciones de coste. La opción recomendada entre las distancias algebraicas es `HDLT_NA` y la función de coste recomendada entre las distancias geométricas es minimizar el error de reproyección.

`tipoFCoste`	Función de Coste
	Distancia Algebraica
0	`HDLT_B`
1	`HDLT_NA`
2	`HDLT_G`
3	`HDLT_NAG`
	Distancia Geométrica
4	Error en la segunda imagen
5	Error de transferencia simétrico
6	Error de reproyección (Gold Standard)

Los algoritmos de optimización que selecciona `tipoOpt` están ordenados: de los más generales a los más específicos (rápidos). No todas las combinaciones con `tipoFCoste` son posibles.

`tipoOpt`	Tipo de Optimización
0	`fminunc`
	Levenberg-Marquardt
1	`LMB`
2	`LMPB`
3	`LMPSI`
4	`LMPSHI_afin`: opción recomendada
5	`lsqnonlin`
6	`LMBChi2`

Se asume que la matriz de covarianzas $\Sigma_{\mathbf{X}}$ es la matriz identidad, para el caso de optimización de alguna de las funciones de coste de distancia geométrica y utilización del algoritmo de Levenberg-Marquardt.

El marco descriptivo común LM aplicado al error en una imagen ha sido descrito en § 4.3.1. La

función de coste está programada en `Coste_Error1img`. La función modelo es `Error1img`. Para el error de transferencia simétrico § 4.3.2, la función modelo es `ErrorTS` y la función de coste se llama `Coste_ErrorTS`.

En el caso del error de reproyección § 4.3.3, la función modelo es `ErrorReproyAfin`, que llama a la rutina `Parametros2MatricesEnReproy` para traducir el vector de parámetros $\mathbf{P} \rightarrow (\mathtt{H}, \hat{\mathbf{x}}_i)$. La función de coste es `Coste_ErrorReproyAfin`. El sufijo "`Afin`" en el nombre de las rutinas recuerda que la contribución de cada punto en cada imagen a $\widehat{\mathbf{X}}$ son 2 dos coordenadas afines y no 3 coordenadas homogéneas. Probé minimizando con $\mathbf{P}$ de dimensión $3n+9$ y $\mathbf{X}$ también en coordenadas homogéneas, $\dim \mathbf{X} = 6n$, y los resultados eran peores que minimizando la distancia geométrica anterior.

En coordenadas homogéneas las fórmulas de $U, V_i, W_i \ldots$ según las recetas de los algoritmos de Levenberg-Marquardt de [1] quedaban más sintéticas que en coordenadas afines, por eso parecían más atractivas. El único problema es que la función de coste no es una distancia geométrica en el plano de la imagen porque $\mathbf{X}$ son coordenadas homogéneas, no afines. Se minimiza la suma de cuadrados de distancias entre coordenadas homogéneas.

Derivadas exactas. La rutina `LMPSHI_afin` calcula la matriz jacobiana exacta del cálculo de Homografías 2D mediante el Gold Standard. Las siglas significan: **L**evenberg - **M**arquardt **P**articionado **S**parse (Disperso) aplicado a **H**omografías suponiendo que la matriz de covarianzas del vector de medidas es la **I**dentidad: $\Sigma_{\mathbf{X}} = \mathrm{Id}$ (Suposición habitual si se desconoce a priori). Es una adaptación del algoritmo A4.3 de [1, pág. 577] y sigue la estructura general del `LMB`.

Optimizaciones. De las siete optimizaciones que se pueden elegir sólo faltan por comentar las dos que utilizan rutinas propias de MATLAB 6. Todavía caben más posibilidades, como utilizar las rutinas del "Numerical Recipes in C" que he traducido a MATLAB, o explotar un poco más las rutinas propias de MATLAB (pasando información de las derivadas...), pero por ahora ya hay suficientes.

La función `fminunc` es la dedicada a la minimización sin restricciones de una función de $\mathbb{R}^M \rightarrow \mathbb{R}$, es decir, que hay que pasarle la función de coste, no el modelo. Se puede utilizar para las tres funciones de coste de distancia geométrica. Todos los parámetros respecto a los cuales minimizar deben estar en un mismo vector o matriz, no más. También se pueden pasar, directamente a través de `fminunc`, argumentos opcionales a la función de coste, como el vector de medidas, $\mathbf{X}$.

Existe la posibilidad de pasar algo parecido a la función modelo, en lugar de la función de coste. La rutina de MATLAB: `lsqnonlin` resuelve problemas de mínimos cuadrados no lineales. Busca

$$\min_{\mathbf{P}} \sum (\mathtt{FUN}(\mathbf{P}))^2$$

Dentro de la rutina se realiza implícitamente el cuadrado y la suma, por lo que hay que proporcionar una función que devuelva el vector $\mathtt{FUN}(\mathbf{P}) = \mathbf{X} - f(\mathbf{P})$. Así,

$$\min_{\mathbf{P}} \sum (\mathtt{FUN}(\mathbf{P}))^2 = \min_{\mathbf{P}} \|\mathbf{X} - f(\mathbf{P})\|^2$$

Tal función es `funReproyAfin` y está programada sólo para implementar el error de reproyección, así que el tipo de optimización 5 sólo vale para dicha función de coste.

La minimización se hace según el algoritmo de Levenberg-Marquardt interno de MATLAB si se indica en las opciones contenidas en `optimset` y pasadas a `lsqnonlin` de la forma:

```
options = optimset('LevenbergMarquardt','on');
```

`H_RANSAC`

Estima homografías del plano entre pares de imágenes mediante la técnica robusta RANSAC, según está explicado en § 4.4.

ENTRADA: una matriz con las correspondencias de puntos entre imágenes $(\mathbf{x}_i, \mathbf{x}'_i)$, en coordenadas homogéneas.

SALIDA:

`Hopt`	matriz de la homografía
`xc`	correspondencias de puntos corregidos en las imágenes $(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i)$
`coste`	mínima distancia geométrica (error de reproyección) de los *inliers*
`maxnuminliers`	numero máximo de *inliers*
`Soporte`	índices de las columnas de `xc` que identifica a los *inliers*
`mindist`	vector de distancias del modelo a los puntos ajustados
`epsilon`	estimación de la proporción de *outliers*

La distancia $d_\perp$ de cada punto al modelo es el error de transferencia simétrico y lo calcula la rutina `Distancia_ErrorTS` .

Otra parte del algoritmo que es mejorable es la selección de los puntos de la muestra aleatoria (a): como se sugiere, se podría dividir la imagen en regiones y tomar en cada muestra puntos suficientemente alejados. El comando `clusterdata` de MATLAB facilita esa agrupación de los puntos en regiones.

D.5. Matriz de rotación

Se han implementado distintas rutinas para manejar con facilidad la información de una rotación del espacio; en concreto, las conversiones entre las distintas representaciones: matriz, cuaterniones, ángulos de Euler, junto con eje y ángulo.

Matriz - eje y ángulo
Las rutinas `VecAngle2Rot` de ADREP-3D y `dof2rotacion` calculan la matriz de rotación dados el eje y el ángulo del movimiento. La rutina `Rot2VecAngle` realiza la función inversa: calcula el eje y el ángulo de una matriz de rotación dada. Los fundamentos matemáticos se detallan en § 5.2.1.

Antes de conocer la fórmula de Rodrigues y la rutina `VecAngle2Rot`, programé `dof2rotacion` , (degrees of freedom to rotación) la cual construye la matriz de rotación según la ecuación 5.1, cuyo único paso complicado es hallar los puntos circulares $\mathbf{I}$ y $\mathbf{J}$. La rutina `Plano2PtosCirculares` es la encargada de hallar esos puntos: dadas las coordenadas duales, $\mathbf{a} = (a_1, a_2, a_3, a_4)^\top$, de un plano en $\mathbb{P}^3$ que pasa por el origen de coordenadas ($a_4 = 0$), halla los dos puntos de corte del plano con la cónica absoluta en la referencia canónica.

Matriz - cuaterniones
La rutina `Quat2Rotation` de ADREP-3D calcula la matriz de rotación asociada a un cuaternión, mientras que `Rotation2Quat` realiza la función inversa: calcula el cuaternión de una matriz de rotación dada.

Cuaterniones - eje y ángulo
Es posible pasar de la representación de cuaternión a la de eje y ángulo de forma sencilla. Las rutinas para esta conversión son: `Quat2VecAngle`, que devuelve el eje y ángulo de rotación asociado a un cuaternión y `VecAngle2Quat`, que realiza la conversión inversa: devuelve el cuaternión asociado un eje y ángulo de rotación.

El cuaternión $\mathbf{q} = (q_x, q_y, q_z, q_w)^\top$ codifica una rotación de ángulo $\theta = 2 \arccos(q_w)$ radianes alrededor del eje de dirección unitaria $\mathbf{u} = \frac{1}{\operatorname{sen}(\arccos(q_w))}(q_x, q_y, q_z)^\top$.

En otras palabras, se puede codificar una rotación de ángulo θ radianes alrededor del eje de dirección unitaria $\mathbf{u} \in \mathbb{R}^3$ mediante el siguiente cuaternión:

$$\mathbf{q} = (q_x, q_y, q_z, q_w)^\top = (\text{sen}(\theta/2)\mathbf{u}, \cos(\theta/2))^\top$$

Matriz - ángulos de Euler
La rutina `AngulosEuler2Rotation` construye una matriz de rotación a partir de los ángulos de Euler, según las fórmulas indicadas en § 5.2.1.

Aproximación de una rotación
También se incluye la rutina `NearestRotMatrix`, que aproxima una matriz por una de rotación utilizando la descomposición en valores singulares.

D.6. Matriz de proyección

Primero se presentan las rutinas relacionadas con la descomposición de la matriz de una cámara proyectiva finita y luego las funciones de MATLAB que implementan los algoritmos de estimación.

Factorización

`CameraMatrix2KRC`

Descompone una matriz de proyección `P` para obtener las matrices `K`, `R` y vector vector $\widetilde{\mathbf{C}}$ tal que $\mathtt{P} = \mathtt{K}[\mathtt{R} \mid \mathbf{t}] = \mathtt{KR}[\mathtt{I} \mid -\widetilde{\mathbf{C}}] = [\mathtt{M} \mid -\mathtt{M}\widetilde{\mathbf{C}}]$ siendo $\mathtt{M} = \mathtt{KR}$. Como referencia se propone el ejemplo 5.2 de [1, pág. 150]. El algoritmo realiza la descomposición RQ de `M` mediante la descomposición QR (comando `qr` de MATLAB).

`CameraMatrix2KRC_RQ`

Es una rutina equivalente a la anterior, que realiza la descomposición RQ de `M` mediante una rutina propia: `DescomposicionRQ`.

`DescomposicionRQ`

REALIZA: la descomposición de una matriz `A` (3×3) de la forma $\mathtt{A} = \mathtt{RQ}$, siendo `R` una matriz triangular superior y `Q` una matriz ortogonal , $\mathtt{Q}^\top\mathtt{Q} = \mathtt{I}$.

Referencia: [1, pág. 552]: A3.1.1 Givens Rotations and RQ decomposition.

$$\mathtt{A} = \mathtt{R}\mathtt{Q}_z^\top\mathtt{Q}_y^\top\mathtt{Q}_x^\top$$

ENTRADA: una matriz `A` (3×3)
SALIDA: las matrices `R`, triangular superior 3×3, `Q` matriz de rotación 3×3 y los tres ángulos de Euler $\theta_x, \theta_y, \theta_z$ que parametrizan la rotación.

`NormalizaKR`

Tras una descomposición RQ o QR de una matriz $\mathtt{M} = \mathtt{KR}$, todavía es necesario realizar un escalado de la matriz `K` devuelta por el algoritmo de descomposición para que represente una afinidad: $\mathtt{K}(3,3) = 1$ (normalización afín) y además unos cambios de signo para que su diagonal principal sea positiva $(\alpha_u, \alpha_v > 0)$. Así `K` es una matriz de parámetros intrínsecos. Las modificaciones afectan a la matriz ortogonal `R` para que se conserve el producto: $\mathtt{M} \sim \mathtt{KR}$.

`KRC2CameraMatrix`

Construye la matriz de proyección $\mathtt{P} = \mathtt{K}[\mathtt{R} \mid -\mathtt{R}\widetilde{\mathbf{C}}]$ a partir de: `K`, `R` y $\widetilde{\mathbf{C}}$. No sólo hace falta descomponer una matriz de proyección euclídea, también hace falta crearla a partir de sus elementos, por ejemplo en el ajuste de haces euclídeo.

Algoritmos de estimación

En este apartado no sólo se incluyen los algoritmos lineal y óptimo descritos en § 5.4, para cámaras proyectivas generales, sino que también han sido programados los respectivos algoritmos para cámaras afines.

`PDLT_B`

REALIZA: la parte de inicialización (lineal) del algoritmo 6.1 de [1, pág. 170] y se le podría denominar "The basic Direct Linear Transformation (DLT) for the Camera Matrix P".

ENTRADA: una matriz, `x`, con las coordenadas homogéneas de los puntos de la imagen, otra matriz, `X`, con los puntos 3D (misma idea que `x`) y dos variables más: `tipoSol`, la cual selecciona el tipo de método empleado para hallar la matriz de proyección estimada (SVD o EIG) y `verb`, que indica la verbosidad (si hay que escribir por pantalla resultados intermedios o no).

SALIDA: la matriz de la proyección `P`, 3×4, el `coste` del ajuste al minimizar la distancia algebraica y $\widetilde{A}$, la matriz de diseño reducida ([1, pág. 175], que cumple $\|\mathtt{A}\mathbf{p}\| = \|\widetilde{A}\mathbf{p}\|$. Sólo se devuelve en caso de `tipoSol`=0 (SVD).

También realiza cuatro comprobaciones previas a cualquier cálculo: mirar si hay puntos sin correspondencia, si hay suficientes puntos para calcular la matriz de proyección (al menos 6), si `x` contiene puntos de $\mathbb{P}^2$ y si `X` contiene puntos de $\mathbb{P}^3$.

El algoritmo lineal con normalización afín de los datos (puntos proyectados y puntos 3D) está programado en: `PDLT_NA`. Es el algoritmo 6.1 de [1, pág. 170], que se conoce como "The normalized Direct Linear Transformation (DLT) for the Camera Matrix P".

Tiene la misma entrada-salida (salvo la matriz de diseño reducida) que `PDLT_B`. Calcula el coste como la distancia algebraica de la matriz de diseño construida con los datos normalizados. No se olvida de desnormalizar el vector solución (matriz de proyección según las inversas de las transformaciones afines aplicadas a los datos). Proporciona un mejor punto de partida de la búsqueda no lineal que `PDLT_B`.

`PADLT`

REALIZA: los pasos (ii) y (iii) del algoritmo 6.2 de [1, pág. 174] y a estos pasos los podríamos llamar: "The basic Direct Linear Transformation (DLT) for an affine Camera Matrix".

ENTRADA: una matriz, `x`, con las coordenadas homogéneas de los puntos de la imagen, otra matriz, `X`, con los puntos 3D (misma estructura que `x`). Deben ser coordenadas homogéneas compatibles con las afines: última coordenada unitaria.

SALIDA: la matriz de la proyección $\mathtt{P}_A$, 3×4 que mejor se ajusta a los datos.

Realiza las mismas cuatro comprobaciones previas que `PDLT_B` y dos más para comprobar que las coordenadas homogéneas son compatibles con las afines. Se utiliza como inicialización del Gold Standard para cámaras afines.

`GoldStandardProyMatrix`

Es algoritmo 6.1 de [1, pág. 170], llamado "The Gold Standard algorithm for estimating `P` from world to image point correspondences in the case that the world points are very accurately known".
ENTRADA:

`x`	coordenadas homogéneas de los puntos de la imagen
`X`	coordenadas homogéneas de los puntos 3D
`tipoOpt`	Selecciona el tipo de optimización de la función de coste

SALIDA:

`P`	matriz de la proyección que minimiza el error de reproyección
`coste`	mínima distancia geométrica alcanzada, con los datos normalizados

A mi juicio veo peligrosa la normalización afín que se realiza sobre los puntos 3D. Esto no tiene peligro en una reconstrucción afín o de semejanza, sin embargo en una reconstrucción proyectiva no tiene sentido realizar una normalización afín de las coordenadas de los puntos del espacio. Se podría implementar un algoritmo que sólo normalice los datos en las imágenes.

Las funciones modelo y de coste se llaman, respectivamente, `ProjErrorReproy` y `Coste_ProjErrorReproy`. La búsqueda del mínimo se puede realizar mediante uno de varios algoritmos de optimización, según el valor de la variable `tipoOpt`. Si no se pasa este argumento a la rutina, se emplea `tipoOpt`=4.

`tipoOpt`	Tipo de Optimización
0	`fminunc`
	Levenberg-Marquardt (LM)
1	`LMBchi2`
2	`LMB`
3	`LMSI`
4	`LMBPI`

Los dos primeros algoritmos de optimización se basan en evaluaciones de la función de coste, mientras que los tres siguientes optimizan mediante la función modelo. Los cuatro primeros tipos de optimización son de propósito general, por lo que si utilizan derivadas, las estiman de forma numérica. La rutina `LMBPI` es la particularización del algoritmo LM para este problema concreto y calcula la matriz jacobiana de forma exacta, a partir del vector de parámetros $\mathbf{P}$ y el vector de medidas estimado $f(\mathbf{P})$, según se indica en § 5.4.2.

`GoldStandardProyMatrixAfin`

El algoritmo Gold Standard para estimar la matriz de proyección afín, $\mathtt{P}_A$. Es algoritmo 6.2 de [1, pág. 174], llamado "The Gold Standard algorithm for estimating an affine Camera Matrix $\mathtt{P}_A$ from world to image point correspondences".

ENTRADA: una matriz, `x`, con las coordenadas homogéneas de los puntos de la imagen, otra matriz, `X`, con los puntos 3D (misma idea que `x`). Deben ser coordenadas homogéneas compatibles con las afines: última coordenada unitaria.

REALIZA: una normalización de los datos mediante `NormalizAfin` (D.3.1), y obtiene la solución mediante el algoritmo lineal `PADLT`; no necesita iterar. Por último desnormaliza la solución.

SALIDA: la matriz de la proyección $\mathtt{P}_A$, 3×4 que minimiza la distancia geométrica del error de reproyección suponiendo cámara afín y el valor del mínimo en `coste`.

D.7. Matriz fundamental

Antes de ver los algoritmos de estimación de `F` comentemos un par de rutinas relacionadas: cómo obtener la matriz fundamental a partir de dos matrices de proyección y cómo calibrar proyectivamente dos cámaras.

Relación de la matriz fundamental `F` con las matrices de proyección

`CameraMatrix2F`

Calcula la matriz fundamental `F` consistente con dos dos matrices de proyección `P` y `P'` dadas, según lo explicado en § 6.2.5.1. En forma concisa:

$$(\mathtt{P}, \mathtt{P}') \rightarrow \mathtt{F}$$

`F2CanonicalCameraMatrix`

Calcula dos matrices de proyección compatibles con una matriz fundamental `F` dada, según lo visto en § 6.2.5.2. De forma resumida:

$$\mathtt{F} \rightarrow (\mathtt{P}, \mathtt{P}')$$

Algoritmos de estimación

`F_7puntos`

Implementa el algoritmo para calcular la matriz fundamental exacta a partir de 7 parejas de puntos, según está descrito en § 6.2.3.1.

ENTRADA:
- `x(3,7,2)` coordenadas homogéneas de las parejas de puntos entre las 2 imágenes.

SALIDA:
- `F(3,3,[1 o 3])` matriz fundamental. Puede haber 1 o 3 soluciones.
- `NumRaicesReales` número de soluciones reales (soluciones del polinomio cúbico (6.7)).

Internamente realiza comprobaciones internas de si las correspondencias pasadas son puntos de $\mathbb{P}^2$, si son entre dos imágenes y si hay 7 puntos.

`FDLT_B`

Implementa el algoritmo de los 8 puntos e imposición a posteriori de la condición de singularidad de `F` ($\det(\mathtt{F}) = 0$), sin normalización afín de los datos.

ENTRADA:
- `P(3,npuntos)` puntos en la primera imagen (coords. homogéneas)
- `Q(3,npuntos)` puntos en la segunda imagen (en el mismo orden de columnas que `P`)
- `tipoSol` selecciona el método empleado para hallar `F`: `svd` o `eig`
- `verb` verbosidad (si hay que escribir por pantalla resultados intermedios o no)

SALIDA:
- `F` matriz fundamental
- `coste` menor valor singular de la matriz de diseño `A`

Si sólo se pasan los dos primeros argumentos de entrada, se decide que se resuelve el sistema mediante la descomposición en valores singulares (SVD) en lugar de mediante autovalores y no se imprime ningún resultado en la ventana de comandos.

Internamente llama a `F2SingularF` de ADREP-3D para imponer a posteriori la condición $\det(\mathtt{F}) = 0$ (proyección de `F` sobre el conjunto de las matrices singulares de 3×3). `FDLT_B` es equivalente a `LinFundMatrix` de ADREP-3D.

`FDLT_NA`

Implementa el algoritmo de los 8 puntos e imposición a posteriori de la condición de singularidad de `F` ($\det(\mathtt{F}) = 0$) y con normalización afín de los datos, tal como se describe en § 10.2 de [1, pág. 265]. Posee los mismos argumentos de entrada y salida que `FDLT_B`.

La normalización afín de los puntos la realiza llamando a la rutina`NormalizAfin` . La justificación de tal normalización la realiza en su artículo [5]. También está recogida en [30], § 3.2.5. Hay una revisión de este algoritmo en [43], donde se da una justificación más de la normalización de los datos, se comparan estadísticamente varios algoritmos normalizados de los 8 puntos, se hace una clasificación de las técnicas de estimación en un marco común, situando entre ellas este algoritmo. Hay otra comparación de métodos de estimación en [44].

`F_MinAlgebraicDistance`

Implementa el algoritmo de § 6.2.3.5, encerrado entre bloques de normalización de los datos y desnormalización de la matriz fundamental, como en la rutina `FDLT_NA`.

ENTRADA:

`x(3,npuntos,2)`	coordenadas homogéneas de las correspondencias de puntos entre imágenes
`tipoOpt`	selecciona el algoritmo de optimización a utilizar
`NumMaxFunEvals`	número máximo de evaluaciones de la función de coste en los algoritmos propios de MATLAB: `fminunc`, `fminsearch`, `lsqnonlin`

SALIDA:

`F`	matriz fundamental que minimiza la distancia algebraica
`coste`	menor valor singular de la matriz de diseño con los datos normalizados

La optimización se puede realizar sobre la función modelo `FundErrorAlgebraic` o sobre la función de coste directamente, `Coste_FundErrorAlgebraic`.

El problema de limitar en $\mathbb{R}^9$ la búsqueda de la matriz fundamental al subespacio generado por las columnas de `E` se resuelve mediante la rutina `ConstrainedMin2`. El epipolo desde el cual iniciar la búsqueda es obtenido mediante `NumKernel` a partir de la estimación inicial de `F` obtenida mediante `FDLT_NA`.

Optimizaciones. Se han programado distintas optimizaciones, dependiendo del valor de `tipoOpt`.

`tipoOpt`	Tipo de Optimización
0	`fminunc`
1	`fminsearch`
	Levenberg-Marquardt
2	`lsqnonlin`
3	`LMB`
4	`LMBchi2`

Las rutinas `fminunc`, `fminsearch` y `LMBchi2` actúan sobre la función de coste, las otras dos, sobre la función modelo suponiendo que el vector de medidas objetivo es el nulo, como aplica en este problema. Dan buenos y rápidos resultados las optimizaciones 0 y 3. El resto tarda un poco más. Ninguna de las opciones calcula derivadas exactas.

`GoldStandardMatrizFund`

Es la implementación del algoritmo explicado en § 6.2.3.6.

ENTRADA:

`x(3,npuntos,2)`	coordenadas homogéneas de las correspondencias de puntos entre imágenes
`tipoOpt`	selecciona el algoritmo de optimización a utilizar

SALIDA:

`F`	matriz fundamental
`coste`	distancia geométrica mínima alcanzada
`xc(3,npuntos,2)`	correspondencias de puntos corregidos en las imágenes $(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i)$
`P(3,4,2)`	matrices de proyección $(\mathtt{P}, \mathtt{P}')$
`X3d(4,npuntos)`	coordenadas homogéneas de los puntos 3D $\hat{\mathbf{X}}_i$ para la distancia geométrica mínima

La función modelo es `FundErrorReproy`, la cual llama internamente a `FundParametros2MatricesEnReproy` para traducir el vector de parámetros. La función de coste es `Coste_FundErrorReproy`.

Optimizaciones. Existen 7 posibles algoritmos de optimización programados, según el valor de `tipoOpt`.

`tipoOpt`	Tipo de Optimización
0	`fminunc`
	Levenberg-Marquardt
1	`LMB`
2	`LMPB`
3	`LMPSI`
4	`LMPSFI`
5	`lsqnonlin`
6	`LMBChi2`

La opción de minimizar la función de coste con `fminunc` no es recomendable, ya que a veces se queda "colgado" MATLAB o no consigue costes muy bajos. Tampoco es recomendable utilizar la opción `LMBchi2`, pues puede tardar mucho en calcular el Hessiano si hay muchos puntos. Las opciones `LMB`, `LMPB`, `LMPSI` y `LMPSFI` dan buenos resultados porque explotan el conocimiento del modelo. De entre ellas la última es más rápida y es la que por defecto está seleccionada. Existe la opción de utilizar la rutina `lsqnonlin` , a la cual hay que pasar la función que devuelve el vector $\mathtt{FUN}(\mathbf{P}) = \mathbf{X} - f(\mathbf{P})$: `funFundReproy`.

En la rutina `LMPSFI` está programado el algoritmo de **L**evenberg - **M**arquardt **P**articionado **S**parse (Disperso) aplicado al cálculo de la matriz **F**undamental suponiendo que la matriz de covarianzas del vector de medidas es la **I**dentidad: $\Sigma_{\mathbf{X}} = \mathrm{Id}$. Es una adaptación del algoritmo A4.3 de [1, pág. 577]. Sigue la estructura general del `LMB`.

`F_RANSAC`

Esta rutina calcula la matriz fundamental de forma robusta, según lo dicho en § 6.2.3.7.

ENTRADA:

`x(3,npuntos,2)`	coordenadas homogéneas de las correspondencias de puntos entre imágenes

SALIDA:

`Fopt`	matriz fundamental
`coste`	mínima distancia geométrica (error de reproyección) de los *inliers*
`xc(3,npuntos,2)`	correspondencias de puntos corregidos en las imágenes $(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i)$
`P(3,4,2)`	matrices de proyección $(\mathtt{P}, \mathtt{P}')$
`X3d(4,npuntos)`	coordenadas homogéneas de los puntos 3D $\hat{\mathbf{X}}_i$
`maxnuminliers`	numero máximo de *inliers*
`Soporte`	índices de las columnas de `xc` que identifica a los *inliers*
`mindist`	vector de distancias del modelo a los puntos ajustados
`epsilon`	estimación de la proporción de *outliers*

La proporción *outliers* estimada es de $\epsilon_{\text{fijo}} = 0{,}2$. Las dimensiones máximas de las imágenes influyen en la desviación típica de ruido esperado, $\sigma = \text{mín}\{\Delta_x, \Delta_y, 10\}$. Para medir el "tamaño" de las imágenes de las que proceden las correspondencias de puntos se utiliza él máximo rango dinámico Δ de las coordenadas x e y de las correspondencias de puntos pasadas. En ningún caso se espera que los puntos sean detectados en las imágenes con más de 10 píxeles de desviación típica de ruido.

`GoldStandardFundAfin`

Esta rutina pertenece al capítulo de geometría epipolar afín de [1] y es el algoritmo 13.1, pág. 340: "The Gold Standard algorithm for estimating $\mathtt{F}_A$ from n images correspondences over 2 views".

ENTRADA:

`x(3,npuntos,2)`	coordenadas homogéneas de las correspondencias de puntos entre imágenes

SALIDA:

`FA`	matriz fundamental afín, $\mathtt{F}_A$

Cámaras afines. En contadas ocasiones en las que la geometría de la escena no presente mucha perspectiva, es posible utilizar la aproximación de cámaras afines en sustitución de las cámaras proyectivas. La cámara afín es una buena aproximación en muchas situaciones prácticas. Su gran ventaja es que debido a su linealidad, es posible resolver muchos de los algoritmos óptimos mediante álgebra lineal, mientras que en el caso de cámaras proyectivas las soluciones involucran polinomios de alto grado (como la triangulación) o sólo es alcanzable a través de optimización numérica (como el Gold standard para la matriz fundamental).

D.8. Triangulación

Este apartado recoge las dos rutinas de triangulación explicadas en § 6.3 y en el capítulo 11 de [1].

`TriangulacionLineal`

ENTRADA:

`x(3,npuntos,2)`	coordenadas homogéneas de las correspondencias de puntos entre imágenes
`P(3,4,2)`	matrices de proyección $(\mathtt{P}, \mathtt{P}')$

SALIDA:

`X(4,npuntos)`	coordenadas homogéneas de los puntos 3D
`costes`	vector con el coste de ajuste de cada punto 3D 3D

La rutina `LinearTriangulation` de ADREP-3D realiza lo mismo, pero sólo para 1 punto.

`TriangulacionOptima`

Implementa el algoritmo 11.1 de [1, pág. 304].

ENTRADA:

`x(3,npuntos,2)`	coordenadas homogéneas de las correspondencias de puntos entre imágenes
`M`	Si `M` es de 3×3, es la matriz fundamental, `F`, entre ambas proyecciones. Si `M` es de $3 \times 4 \times 2$, son las matrices de proyección `P`.

SALIDA:

`X(4,npuntos)`	coordenadas homogéneas de los puntos 3D
`xe(3,npuntos,2)`	coordenadas homogéneas de los puntos proyectados corregidos.

Esta función se utiliza para proporcionar una inicialización del Gold Standard para `F`, en el cual se estima a la vez la matriz fundamental y los puntos 3D $\mathbf{X}_i$ a partir de las correspondencias de puntos en las imágenes $\mathbf{x}_i$ y $\mathbf{x}'_i$.

La rutina `OptimalTriangulation` de ADREP-3D realiza lo mismo, pero sólo para 1 punto y pasándole la matriz fundamental .

D.9. Tensor trifocal

Primero se indicarán las rutinas para construir el tensor trifocal a partir de las matrices de proyección y viceversa: extraer las matrices de proyección de un tensor trifocal dado. A continuación se verán las rutinas que he programado sobre los algoritmos descritos en § 6.4.

Relación del tensor trifocal $\mathcal{T}$ con las matrices de proyección

`P2Trifocal`

Dadas tres matrices de proyección `P` $=$ `A`, `P`$'$ $=$ `B` y `P`$''$ $=$ `C`, calcula el tensor trifocal $\mathcal{T}$ asociado a la primera matriz de proyección. Se basa en la fórmula (6.13). En este caso no hace falta estimar $\mathcal{T}$, ya que se construye de forma directa a través de las matrices de proyección. Una forma concisa de expresarlo:

$$(\mathtt{P}, \mathtt{P}', \mathtt{P}'') \rightarrow \mathcal{T}$$

`Trifocal2F_P`

Calcula tres matrices de proyección compatibles con un tensor trifocal $\mathcal{T}$ dado, según § 6.4.4. De forma concisa:

$$\mathcal{T} \rightarrow (\mathtt{P}, \mathtt{P}', \mathtt{P}'')$$

Utilizando la notación matricial para el tensor trifocal, los productos del tensor por un vector, del tipo $[\mathtt{T}_1, \mathtt{T}_2, \mathtt{T}_3]\mathbf{e}$ y $[\mathtt{T}_1^\top, \mathtt{T}_2^\top, \mathtt{T}_3^\top]\mathbf{e}$ se calculan mediante la rutina `TrifocalxVect`.

Algoritmos de estimación

`TrifocalDLT_B`

Implementa el algoritmo lineal para estimar el tensor trifocal a partir de correspondencias de puntos y/o correspondencias de rectas, sin normalización de los datos, § 6.4.3.1. Referencia: paso (*iii*) del algoritmo 15.1 de [1, pág. 383].

No utiliza correspondencias mixtas puntos-rectas. Como no se impone ninguna restricción sobre los elementos del tensor, la solución puede no ser un tensor trifocal válido.

ENTRADA: matriz `x` con las correspondencias de puntos entre imágenes (x, x', x'') y/o una matriz `r` con las correspondencias de rectas entre imágenes (l, l', l''), ambas en coordenadas homogéneas.

REALIZA: Para cada correspondencia de puntos llama a la rutina `pto_pto_pto`, obtiene las 9 ecuaciones de esa correspondencia, y las añade a la matriz de diseño, `A`. Para cada correspondencia de rectas llama a la rutina `line_line_line`, obtiene las 3 ecuaciones de esa correspondencia, y las añade a la matriz de diseño , `A`. Esta matriz es de tamaño $(9n + 3\,\text{nrectas}) \times 27$ y tiene cierta estructura dispersa.

SALIDA:

`Ti`	tensor trifocal
`coste`	menor valor singular de la matriz de diseño, es decir, la distancia algebraica
`A_red`	matriz de medida reducida
`A`	matriz de diseño

Observación. Si no se utilizan rectas, como mínimo hacen falta 7 puntos, por lo que `A` es de 63×27; se utiliza la SVD "barata" de MATLAB (que sólo calcula las 27 primeras columnas de `U`) y no perdemos información del coste ni el vector `t` solución. No se han tomado 4 ecuaciones linealmente independientes por correspondencia de puntos puesto que computacionalmente no es mucho más costoso hallar la SVD de una matriz 9npuntos $\times$ 27 que una 4npuntos $\times$ 27 y además tampoco es recomendable quitar ecuaciones, por si hay situaciones degeneradas. Lo mismo es aplicable a las correspondencias de rectas, por cada una de ellas se incluyen tres filas en la matriz de diseño, aunque sólo dos sean linealmente independientes.

`TrifocalDLT_NA`

Es la programación del algoritmo lineal para calcular el tensor trifocal a partir de correspondencias de puntos y/o correspondencias de rectas, con normalización afín de los datos. Es el algoritmo 15.1 de [1, pág. 383]. Posee los mismos argumentos de entrada y salida que `TrifocalDLT_B`.

La normalización afín de los puntos y/o las rectas se realiza llamando a la rutina `NormalizAfin` en caso de tener dos o más correspondencias de puntos, si no se utiliza una matriz de normalización prefijada para tratar de equilibrar las coordenadas homogéneas. La desnormalización del tensor trifocal la realiza la rutina `TensorTransformationRule`.

Este algoritmo tampoco considera las restricciones que deberían aplicarse al tensor estimado. Dichas restricciones deberían imponerse antes de la desnormalización, o mejor aún, en la propia estimación SVD del tensor, como se verá en el siguiente método.

`Trifocal_MinAlgebraicDistance`

ENTRADA:

`x(3,npuntos,3)`	correspondencias de puntos entre imágenes (x, x', x''), en coords. homogéneas
`tipoOpt`	selecciona el algoritmo de optimización a utilizar
`NumMaxFunEvals`	número máximo de evaluaciones de la función de coste en los algoritmos propios de MATLAB: `fminunc`, `fminsearch`, `lsqnonlin`
`r`	correspondencias de rectas entre imágenes (l, l', l''), en coords. homogéneas

SALIDA:

`Ti`	tensor trifocal geométricamente válido
`coste`	menor valor singular de la matriz de diseño, es decir, la distancia algebraica

La función de coste es `Coste_TrifocalErrorAlgebraic`. La función modelo `TrifocalErrorAlgebraic` llama internamente a `epipolos2E` para construir la matriz de restricción`E`, de 27×18, a partir de los epipolos. El problema de limitar en $\mathbb{R}^{27}$, la búsqueda del tensor trifocal al subespacio generado por las

columnas de `E` se resuelve mediante la rutina `ConstrainedMin2`. Los epipolos son obtenidos mediante `Trifocal2epipolos` a partir de una estimación del tensor.

Optimizaciones. Se han programado distintas optimizaciones, dependiendo del valor de `tipoOpt`.

`tipoOpt`	Tipo de Optimización
0	`fminunc`
1	`fminsearch`
	Levenberg-Marquardt
2	`lsqnonlin`
3	`LMB`
4	`LMBchi2`

Dan buenos y rápidos resultados las optimizaciones 0 y 3. El resto tarda un poco más. Como `fminunc` es de las mejores y no da problemas, es la opción recomendada.

`GoldStandardTensorTrifocal`

Es la programación del algoritmo de § 6.4.3.3, o el algoritmo 15.3 de [1, pág. 386].

ENTRADA:

`x(3,npuntos,3)`	correspondencias de puntos entre imágenes (x, x', x''), en coords. homogéneas

SALIDA:

`Ti`	tensor trifocal geométricamente válido
`coste`	distancia geométrica mínima alcanzada
`xc(3,npuntos,3)`	puntos corregidos en las imágenes
`P(3,4,3)`	matrices de proyección
`X3d(4,npuntos)`	coordenadas homogéneas de los puntos 3D que minimizan la distancia geométrica

La función de coste `Coste_TrifocalErrorReproy` calcula la distancia geométrica del error de reproyección llamando internamente a la función modelo `TrifocalErrorReproy` para calcular la estimación $f(\mathbf{P})$.

Inicialización.

- La rutina `Trifocal_MinAlgebraicDistance` se utiliza para generar una estimación inicial de un tensor trifocal válido, evaluando un máximo de 50 veces la función de coste de distancia algebraica y con `tipoOpt=1` (`fminunc`), que es bastante rápido. Bastaría con evaluar una sola vez para obtener un tensor trifocal válido, definido por sus matrices de proyección.

- Hay que llevar cuidado en el paso (ii b) de estimación de los puntos 3D mediante la triangulación óptima. Se debe tener en cuenta que si pasamos `F(:,:,1)` a `TriangulacionOptima`, las matrices de proyección en forma canónica calculadas dentro no coincidirán con las ya halladas, `P`. Existe un cambio de referencia que relaciona las dos parejas de matrices de proyección y las coordenadas de los puntos 3D.

  ```
  [X3d2,xe] = TriangulacionOptima(x(:,:,1:2),F(:,:,1));
  H=ChangeBaseMatrix_X2Y(X3d2(:,1:5),X3d(:,1:5));
  ```

  Se verifica la proporcionalidad `X3d` $\sim$ `inv(H)*X3d2`.

  Conclusión. Los puntos 3D y las matrices de proyección siempre van juntos porque una misma proyección se puede obtener de muchas formas, proyectivamente equivalentes: $\mathbf{x} = \mathtt{P}\mathbf{X} = (\mathtt{P}\mathtt{H})(\mathtt{H}^{-1}\mathbf{X})$.

En la práctica, se utiliza el algoritmo `TriangulacionLineal`, aunque se ha modificado la rutina `TriangulacionOptima` para que pueda recibir la matriz fundamental o dos matrices de proyección de una calibración proyectiva y también se podría utilizar en este caso.

- No hace falta realizar el apartado (ii c) según [1], ya que no se utilizan $\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i, \hat{\mathbf{x}}''_i$ para nada en la estimación inicial.

Optimizaciones. Hay 9 posibles optimizaciones programadas:

`tipoOpt`	Tipo de Optimización
0	`fminunc`
	Levenberg-Marquardt
1	`LMB`
2	`LMPB`
3	`LMPSI`
4	`LMPSTI`
5	`LMSI`
6	`lsqnonlin`
7	`LMBChi2`
8	`fminsearch`
9	`xamebsa` (Simulated Annealing)

La opción recomendada es utilizar la rutina `LMPSTI` ya que construye la matriz jacobiana de forma exacta a partir de los datos, no numéricamente. La siguiente recomendada es `LMSI`, en la cual MATLAB aprovecha la estructura dispersa mediante su *sparse toolbox*. Cualquiera de las opciones 1 a 5 obtienen resultados parecidos.

La opción de minimizar la función de coste con `fminunc` no es recomendable. Tampoco es recomendable utilizar la opción `LMBchi2`, pues puede tardar mucho en calcular el Hessiano. Existe la opción de utilizar la rutina `lsqnonlin`, a la cual se debe proporcionar el nombre de la función que devuelve el vector de error `FUN`$(\mathbf{P}) = \mathbf{X} - f(\mathbf{P})$: `funTrifocalReproy`. También se ofrece la posibilidad de optimizar con otras rutinas basadas en simplex's, aunque tardan más y no suelen alcanzar un coste tan bajo como las rutinas LM: `fminsearch` implementa el Downhill Simplex Method de Nelder y Mead (determinista) y `xamebsa` implementa la versión estocástica (enfriamiento simulado).

La rutina `LMPSTI` realiza el algoritmo A4.3 de [1, pág. 577], (`LMPSI`) incluyendo la modificación de calculo exacto de la matriz jacobiana. Sigue la estructura general del `LMB`.

`RecProy_6puntos`

Calcula una reconstrucción proyectiva en $m \geq 3$ imágenes o proyecciones a partir de 6 correspondencias de puntos. Es importante resaltar que sirve para los dos casos: 3 cámaras o más de 3 cámaras. Referencia: § 19.2.4 y § 19.2.5 de [1, pág. 492-493]. Se basa en la dualidad Carlsson-Weinshall.

ENTRADA: una matriz con las correspondencias de 6 puntos entre las m imágenes, en coordenadas homogéneas.

SALIDA: los elementos de una calibración proyectiva,

`P(3,4,ncam,NumRaicesReales)`	matrices de proyección
s `X3d(4,6,NumRaicesReales)`	puntos 3D de la reconstrucción proyectiva

Si $m = 3$, hay una o tres posibles soluciones reales `NumRaicesReales`. En cambio, si $m > 3$, sólo hay una solución. Utiliza las funciones `SeleccionaBase` y `GeneralChangeBaseMatrix`, junto con otras subfunciones o funciones locales.

`Trifocal_exacto`

Construye un tensor trifocal válido a partir de las matrices de proyección devueltas por `RecProy_6puntos`, para cada una de las soluciones reales halladas.

ENTRADA: una matriz con las correspondencias de 6 puntos entre las 3 imágenes, en coordenadas homogéneas.

SALIDA: los elementos de una calibración proyectiva,

`Ti(3,3,3,NumRaicesReales)`	tensor(es) trifocal(es)
`P(3,4,ncam,NumRaicesReales)`	matrices de proyección
`X3d(4,6,NumRaicesReales)`	puntos 3D de la reconstrucción proyectiva
`NumRaicesReales`	1 o 3, porque hay 3 cámaras

`Trifocal_RANSAC`

Calcula el tensor trifocal mediante el algoritmo robusto explicado en § 6.4.3.5. La proporción *outliers* estimada es de $\epsilon_{\text{fijo}} = 0{,}2$ y la desviación típica de ruido esperado es $\sigma = \text{mín}\{\Delta_x, \Delta_y, 10\}$, según lo explicado para la rutina `F_RANSAC`.

ENTRADA: una matriz con las correspondencias de puntos entre imágenes $(\mathbf{x}_i, \mathbf{x}'_i, \mathbf{x}''_i)$, en coordenadas homogéneas.

SALIDA:

`Ti_opt`	tensor trifocal
`coste`	mínima distancia geométrica (error de reproyección) de los *inliers*
`xc`	correspondencias de puntos corregidos en las imágenes $(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i)$
`P_opt`	matrices de proyección $(\mathtt{P}, \mathtt{P}')$
`X3dh_opt`	coordenadas homogéneas de los puntos 3D $\hat{\mathbf{X}}_i$
`maxnuminliers`	numero máximo de *inliers*
`Soporte`	índices de las columnas de `xc` que identifica a los *inliers*
`mindist`	vector de distancias del modelo a los puntos ajustados
`epsilon`	estimación de la proporción de *outliers*

La distancia $d_\perp$ de cada punto al modelo la calcula la rutina `Distancia_TrifocalErrorReproy`. Como su nombre indica, calcula el error de reproyección .

D.10. Tensor cuadrifocal

En esta sección se describen las rutinas relacionadas con la estimación del tensor cuadrifocal y la calibración proyectiva de las 4 cámaras que encierra.

`P2Quadrifocal`

Esta rutina no estima el tenor cuadrifocal, sino que lo construye dadas 4 matrices de proyección, siguiendo la fórmula (6.17). También sirve para calcular un tensor cuadrifocal reducido de forma eficiente: sólo calcula los 36 determinantes no nulos.

`QuadrifocalDLT_B`

Implementa el algoritmo lineal de § 6.5.2.1 para calcular el tensor cuadrifocal a partir de 6 o más correspondencias de puntos en 4 imágenes o proyecciones. Sin embargo, no normaliza los datos (puntos proyectados) ni el tensor estimado es geométricamente válido, ya que no se imponen las restricciones

propias. Realiza los pasos básicos

1. Construir la matriz de diseño `A` del sistema (6.18).
2. Hallar la descomposición en valores singulares de ésta: $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^\top$.
3. Calcular el tensor cuadrifocal que minimiza la distancia algebraica: $\mathbf{q}$ es la última columna de `V`.
4. Calcular el coste o distancia algebraica mínima: el menor valor singular de `A`.
5. Construir la matriz de medida reducida: $\hat{\mathtt{A}}$, para una posible minimización iterativa posterior.

Cada correspondencia de puntos da lugar a 81 ecuaciones lineales (filas de la matriz `A`). La rutina `pto_pto_pto_pto` calcula la matriz $\mathtt{A}_i$ con las 81 ecuaciones de una correspondencia de puntos. La matriz `A` se obtiene juntando al menos 6 matrices $\mathtt{A}_i$ verticalmente.

En la implementación concreta en MATLAB, el convenio que se ha elegido para pasar de la estructura de datos vector $\mathbf{q}$ a la estructura matriz de 4 dimensiones Q (tensor cuadrifocal) es seguir el camino más cómodo y rápido: `Q=reshape(q,3,3,3,3);` y `q=Q(:);`. Con este convenio, la matriz $\mathtt{A}_i$ tiene la asombrosa estructura dispersa de la figura D.2. Los puntos negros indican donde las ecuaciones tienen coeficientes no nulos, en principio. Esas imágenes recuerdan el fractal de Koch. Se distinguen 4 niveles de recursividad en `A`, los mismos que índices del tensor cuadrifocal.

`QuadrifocalDLT_NA`

Es la programación del algoritmo lineal de § 6.5.2.1 para calcular el tensor cuadrifocal a partir de 6 o más correspondencias de puntos en 4 imágenes o proyecciones, normalizando los datos (puntos proyectados). Pero el tensor estimado no es geométricamente válido, ya que no se imponen las restricciones propias. Realiza una normalización afín previa a la llamada a `QuadrifocalDLT_B` y otro posterior de desnormalización.

No se ha encontrado en la literatura la forma de desnormalizar un tensor cuadrifocal que no sea reducido, así que esta rutina sólo informa de cuál es la mínima distancia algebraica con datos normalizados si se deja variar el vector con los elementos de Q en todo el ambiente $\mathbb{R}^{81}$. Este número sirve para compararlo con los que dan las rutinas de minimización iterativa del error algebraico (costes mayores que éste).

`Quadrifocal_Heyden`

Implementa el algoritmo lineal de Heyden de estimación del tensor cuadrifocal y las matrices de proyección, descrito en § 6.5.2.2. Se ha incluido el algoritmo dentro de unos bloques de normalización y desnormalización afín de los puntos proyectados y los resultados obtenidos, respectivamente, para que numéricamente sea mejor (como en el algoritmo de los ocho puntos [5]).

Una vez estimado el tensor cuadrifocal reducido, se utiliza la función `Q_red2Q` para obtener el tensor cuadrifocal completo y sin normalización afín, todo en un único paso mediante la ecuación (6.22), pero combinando las dos transformaciones `T` y `H` en una sola.

$$\mathtt{P}_{\mathbf{x}} = \mathtt{H}^{-1}\mathtt{T}\hat{\mathtt{P}} = \mathtt{C}\hat{\mathtt{P}} \Longrightarrow Q^{ijkl} = \hat{Q}^{abcd}\mathtt{C}_a^i\mathtt{C}_b^{\prime j}\mathtt{C}_c^{\prime\prime k}\mathtt{C}_d^{\prime\prime\prime l}$$

ENTRADA:

`x(3,npuntos,4)` correspondencias de puntos entre las 4 imágenes, en coordenadas homogéneas

SALIDA:

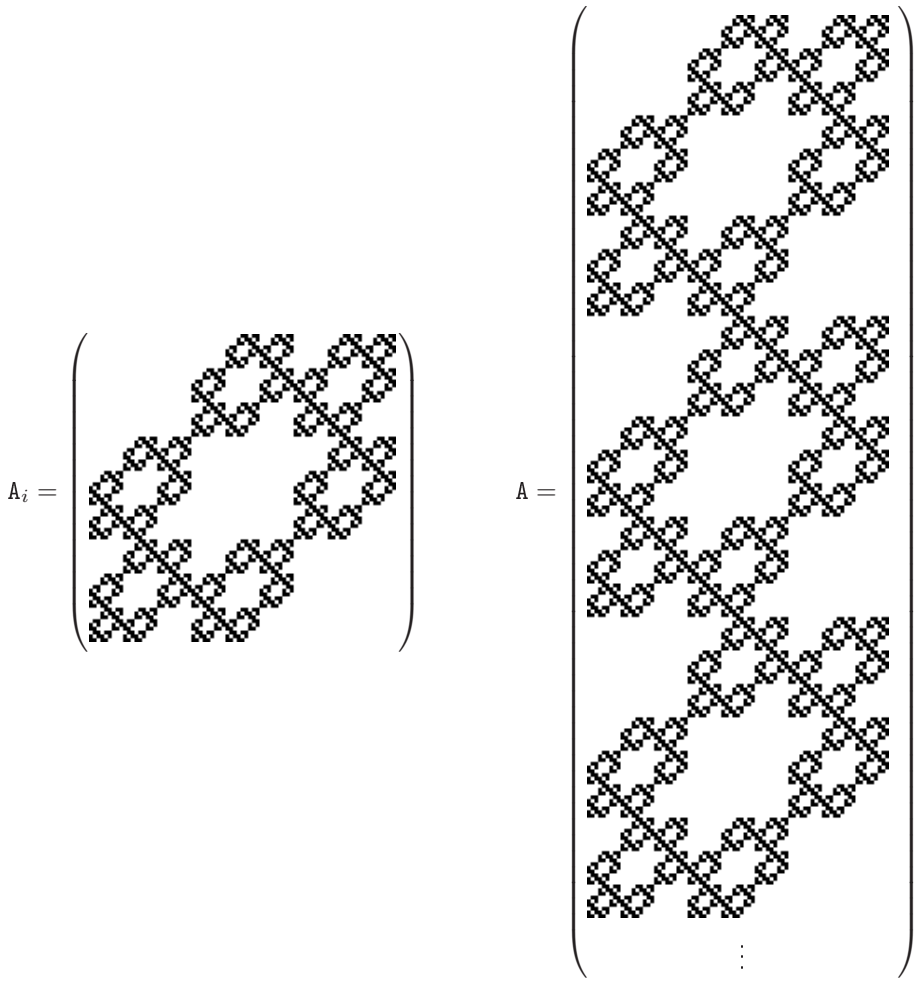


Figura D.2: Estructura de la matriz de diseño `A` y submatrices $\mathtt{A}_i$ de cada correspondencia de puntos.

`Q(3,3,3,3)`	tensor Cuadrifocal
`coste`	menor valor singular de la matriz de diseño
`P(3,4,4)`	matrices de proyección consistentes con el tensor

`Quadrifocal_MinAlgDist_1iter`

Estima el tensor cuadrifocal y las matrices de proyección siguiendo el algoritmo lineal de § 6.5.2.3.

El método descrito, trata de encontrar el vector $\mathbf{q}$ que minimiza la distancia algebraica a la vez que proviene de una configuración de 4 cámaras. Es decir, buscar el mínimo dentro de la variedad de los tensores cuadrifocales *geométricamente válidos*, contenida en $\mathbb{R}^{81}$. Para ello utiliza el algoritmo de minimización con restricciones lineales programado en la rutina `ConstrainedMin2`.

ENTRADA:

`x(3,npuntos,4)`	correspondencias de puntos entre las 4 imágenes, en coordenadas homogéneas

SALIDA:

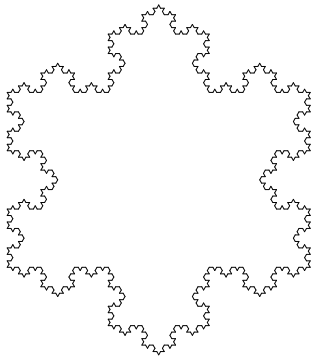


Figura D.3: Fractal de Koch de 6 puntas y 4 niveles

`Q(3,3,3,3)`	tensor Cuadrifocal
`coste`	menor valor singular de la matriz de diseño con los datos normalizados y cumpliendo las restricciones
`P(3,4,4)`	matrices de proyección consistentes con el tensor

Internamente llama a otras rutinas para realizar pasos intermedios:

- `QuadrifocalRestrictionMatrix` construye la matriz de restricciones `E` 81×36 tal que $\mathbf{q} = \mathtt{E}\hat{\mathbf{q}}$ a partir de las transformaciones $\mathtt{T}, \mathtt{T}', \ldots$, siendo $\hat{\mathbf{q}} \in \mathbb{R}^{36}$ el vector con el tensor cuadrifocal reducido.
- `Q_red2a` obtiene el vector $\mathbf{a} \in \mathbb{R}^{18}$ con las entradas diagonales de las matrices de proyección 2 a 4, dado un tensor cuadrifocal reducido $\hat{Q}$, mediante la resolución de las ecuaciones (6.23).
- `a2M` calcula la matriz `M` tal que $\hat{\mathbf{q}} = \mathtt{M}\mathbf{a}'$, construida a partir de $\mathbf{a}$.
- `aap2P` construye las matrices de proyección reducidas de (6.21) dados los vectores $\mathbf{a}$ y $\mathbf{a}'$.

Las rutinas `qp2Q_red` y `Q_red2qp` sirven para convertir la estructura de almacenamiento del tensor cuadrifocal, de vector en array 4D y viceversa, respectivamente.

`Quadrifocal_MinAlgDistR9`

Estima el tensor cuadrifocal y las matrices de proyección siguiendo el primer algoritmo iterativo de § 6.5.2.4. Utiliza la función modelo $f : \mathbb{R}^9 \mapsto \mathbb{R}^{81}$.

ENTRADA:

`x(3,npuntos,4)`	correspondencias de puntos entre las 4 imágenes, en coordenadas homogéneas
`tipoOpt`	selecciona el algoritmo de optimización a utilizar
`NumMaxFunEvals`	número máximo de evaluaciones de la función de coste en los algoritmos propios de MATLAB: `fminunc`, `fminsearch`, `lsqnonlin`

SALIDA:

`Q(3,3,3,3)`	tensor Cuadrifocal
`coste`	menor valor singular de la matriz de diseño con los datos normalizados y cumpliendo las restricciones
`P(3,4,4)`	matrices de proyección consistentes con el tensor

`Quadrifocal_MinAlgDistR27`

Calcula el tensor cuadrifocal y las matrices de proyección mediante el segundo algoritmo iterativo de § 6.5.2.4, es decir, la función modelo $f : \mathbb{R}^{27} \mapsto \mathbb{R}^{81}$. Tiene la misma entrada y salida que `Quadrifocal_MinAlgDistR9`. Es la segunda mejor, en estimación, por detrás del ajuste de haces para la minimización del error Geométrico § 6.5.2.5 (Gold Standard).

Para verificar la relación 6.16 se ha programado la función `CosteAlgP81`, que calcula la suma de cuadrados del tensor $u^i u'^j u''^k u'''^l \epsilon_{ipw} \epsilon_{jqx} \epsilon_{kry} \epsilon_{lsz} Q^{pqrs}$, dados un tensor Q y unas correspondencias de puntos $\mathbf{u}, \mathbf{u}', \mathbf{u}''$ y $\mathbf{u}'''$.

D.11. Geometría multicámara

D.11.1. Reconstrucción proyectiva inicial

Las rutinas de este apartado se refieren a los algoritmos expuestos en § 6.6.3 sobre la obtención de una calibración proyectiva inicial de varias cámaras. Han sido programadas varias rutinas:

`InicNViewLinProjMatrix`

Realiza la calibración proyectiva inicial de varias cámaras según el primer método en § 6.6.3.

ENTRADA:

`x(3,npuntos,ncam)`	coords. homogéneas de los puntos observados en las imágenes

SALIDA:

`P(3,4,ncam)`	matrices de proyección
`xc(3,npuntos,ncam)`	coords. homogéneas de los puntos corregidos en las imágenes
`X3d`	coordenadas homogéneas de los puntos 3D
`F(3,3)`	matriz fundamental de la primera a la última cámara

En realidad los únicos resultados que interesan son `P` y `X3d`. El resultado en `xc` no es fiable, es mejor proyectar los puntos `X3d` mediante `P`.

`InicNView`

Calcula una calibración proyectiva inicial de varias cámaras según el cuarto método en § 6.6.3, pero con una ligera modificación respecto del guión que allí se cita: en lugar de calcular la matriz fundamental entre cada par de cámaras seguidas se calculan las matrices fundamentales de todas las cámaras con la primera.

Como la calibración proyectiva de dos cámaras se realiza mediante la rutina `GoldStandardMatrizFund`, la cual también devuelve los puntos corregidos en ambas imágenes $\hat{\mathbf{x}}_j^i$, se guardan estos valores y se devuelven en la variable de salida xc, aunque no hay que fiarse de ellos.

La matriz del cambio de referencia se construye mediante la rutina `ChangeBaseMatrix_X2Y`, que utiliza dos llamadas a `ChangeBaseMatrix`: una para pasar de la referencia $\mathcal{R}^*$ a la canónica y otra para pasar de ésta a $\mathcal{R}$.

`InicNViewSecuencia`

Estima una calibración proyectiva inicial de varias cámaras según el cuarto método en § 6.6.3, que está suficientemente detallado en el guión.

No hay ningún problema si el número de cámaras es par. En cambio, si hay un número impar $2q+1$ de cámaras, con las $2q$ primeras se sigue el guión anterior, y en un último paso se halla la matriz fundamental de la última imagen con la penúltima, se obtiene el cambio de referencia y se modifica sólo la segunda matriz de proyección (la de la última cámara).

También se realiza la calibración proyectiva de las dos cámaras mediante el algoritmo Gold Standard para `F`. La matriz del cambio de referencia también se construye mediante la rutina `ChangeBaseMatrix_X2Y`: $\mathcal{R}^* \longrightarrow$ canónica $\longrightarrow \mathcal{R}$.

La diferencia y posible ventaja de esta rutina respecto a `InicNView` es que se calculan menos matrices fundamentales.

`InicNViewTrifocal`

Estima una calibración proyectiva inicial de varias cámaras según el quinto método en § 6.6.3. Esta rutina se programó para comparar los resultados con las dos rutinas anteriores, sin embargo los resultados no son significativamente mejores y dado que se tarda más en estimar el tensor trifocal que la matriz de proyección y que sus resultados sólo se necesita como punto inicial de la búsqueda de la calibración proyectiva óptima, no se utiliza regularmente.

D.11.2. Ajuste de haces proyectivo

`BundleAdjustment`

Realiza un ajuste de haces proyectivo de varias cámaras y puntos 3D a partir de una calibración proyectiva inicial y las coordenadas de los puntos observados. Esto lo encapsulado para que sea cómodo de utilizar: a partir de los datos, ofrece distintos caminos de optimización.

Una precaución que se tomó al programar la rutina es que se utilizan coordenadas afines en lugar de coordenadas homogéneas en los argumentos de entrada, así no hay peligro de cometer error en la parametrización de los puntos 3D.

ENTRADA:

`P(3,4,ncam)`	matrices de proyección
`X3d(3,npuntos)`	coordenadas afines de los puntos 3D
`x(2,npuntos,ncam)`	coordenadas afines de los puntos proyectados
`tipoOpt`	tipo de optimización

SALIDA:

`X3dmin(3,npuntos)`	coordenadas afines de los puntos 3D
`pmin(3,4,ncam)`	matrices de proyección
`xc(2,npuntos,ncam)`	coordenadas afines de los puntos proyectados corregidos
`coste`	distancia geométrica mínima alcanzada, medida en píxeles

La función de coste `Coste_BundleAdjustment` calcula el error de reproyección de los npuntos puntos en las ncam imágenes llamando a la función modelo `BundleErrorReproy`. Ésta última llama a la rutina `BundleP2Matrices` para traducir el vector de parámetros a sus elementos $\mathbf{P} \rightarrow (\mathtt{P}^i, \ldots, \mathbf{X}_{aj})$.

Optimizaciones. Según el valor de la variable `tipoOpt` se selecciona uno de los siguientes algoritmos, ordenados de los más generales a los más específicos (rápidos) y por último nuevas incorporaciones.

tipoOpt	Tipo de Optimización
0	fminunc
	Levenberg-Marquardt
1	LMB
2	LMPB
3	LMPSI
4	LMPSBAI
5	lsqnonlin
6	LMBchi2
7	fminsearch

La opción recomendada es la 4, `LMPSBAI`, que calcula las derivadas exactas de la función modelo. Si no se indica un valor para `tipoOPt`, se escoge este camino. Las siglas de `LMPSBAI` significan algoritmo de **L**evenberg-**M**arquardt **P**articionado Disperso (**S**parse) adaptado al **B**undle **A**djustement, con $\Sigma_{\mathbf{X}} = \mathrm{Id}$. Está explicado en [1, pág. 582]. Es un algoritmo doblemente disperso.

La optimización mediante `lsqnonlin` se realiza sobre la función `funBundleReproy`, que es la que devuelve el vector $\mathtt{FUN}(\mathbf{P}) = \mathbf{X} - f(\mathbf{P})$.

Se ha incluido la posibilidad de utilizar el comando `fminsearch`, que implementa el método del Simplex Cuesta Abajo (Downhill Simplex Method § C.3.1), interno de MATLAB. Al igual que `fminunc` y `LMBchi2`, sólo necesita conocer la función de coste, y por lo general es más lento que utilizar información del modelo o su gradiente, tanto más cuanto mayor sea la dimensión del vector de medidas.

`FactorizacionAfin`

El algoritmo de ajuste de haces equivalente, pero para cámaras afines es el algoritmo de factorización para determinar la estimación de máxima verosimilitud en una reconstrucción afín de m cámaras a partir de n correspondencias entre sus imágenes, bajo la suposición de ruido gaussiano habitual. En [1] lo explican en el algoritmo 17.1 pág 426. Es un algoritmo no iterativo que se resuelve mediante la SVD el siguiente problema:

Dadas $n \geq 4$ correspondencias de puntos en m imágenes, $\mathbf{x}_j^i, j = 1, \ldots, n;\ i = 1, \ldots, m$, determinar las matrices de proyección afines $\{\mathtt{M}_i, \mathbf{t}_i\}$ y puntos 3D $\{\mathbf{X}_j\}$ tales que minimizan el error de reproyección

$$\sum_{i,j} (\mathbf{x}_{aj}^i - (\mathtt{M}^i \widetilde{\mathbf{X}}_j + \widetilde{\mathbf{t}}^i))^2$$

donde $\mathtt{M}^i$ es una matriz 2×3, $\widetilde{\mathbf{X}}_j$ es un vector de 3 componentes, $\mathbf{x}_{aj}^i = (x_j^i, y_j^i)^\top$ son las coordenadas afines de las observaciones y $\widetilde{\mathbf{t}}^i$ es un vector de dos componentes.

ENTRADA:

`x` coordenadas afines de las correspondencias de puntos en las imágenes

SALIDA:

`M` partes afines de las matrices de proyección
`t` vectores de traslación a los centroides
`X` coordenadas afines de los puntos 3D reconstruidos

D.11.3. Ajuste de haces proyectivo con distorsión radial

`BundleAdjustmentRadialDist`

Realiza un ajuste de haces proyectivo de varias cámaras y puntos 3D a partir de una calibración proyectiva inicial y las coordenadas de los puntos observados. El modelo de proyección ahora incluye

una etapa no lineal de distorsión radial. Todo se encapsula para que sea fácil de utilizar.

ENTRADA:

`P(3,4,ncam)`	matrices de proyección
`X3d(3,npuntos)`	coordenadas afines de los puntos 3D
`x(2,npuntos,ncam)`	coordenadas afines de los puntos proyectados
`tipoOpt`	tipo de optimización
`M`	número de coeficientes del polinomio de distorsión radial `ncoef` $= 3 \Rightarrow L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3$ o valores iniciales de los parámetros

SALIDA:

`X3dmin(3,npuntos)`	coordenadas afines de los puntos 3D
`pmin(3,4,ncam)`	matrices de proyección
`xc(2,npuntos,ncam)`	coordenadas afines de los puntos proyectados corregidos
`coste`	distancia geométrica mínima alcanzada, medida en píxeles
`kdmin(ncoef+2,ncam)`	parámetros de distorsión radial de cada cámara

Si `M` es un número, se interpreta como el valor ncoef, los parámetros iniciales de la distorsión radial para la búsqueda no se ponen a cero para evitar que la matriz jacobiana salga nula en las columnas de los coeficientes de distorsión. En realidad creo que bastaría con dar un valor inicial a los κ_1^i, aunque los centros de distorsión y resto de coeficientes fuesen cero. En la práctica se da un valor aleatorio a los centros de distorsión, dentro del cuadrado de lado 5 píxeles y centrado en el $(0, 0)$. También se inicializan los κ_1^i a un valor aleatorio entre $\pm 5 \cdot 10^{-6}$. Por otro lado, si `M` es una matriz, se interpreta como valores iniciales de los parámetros de distorsión radial.

La función de coste `Coste_BARadialDist` calcula el error de reproyección de los npuntos puntos en las ncam imágenes llamando a la función modelo `BARadialDist_ErrorReproy`. La función modelo realiza los siguientes pasos:

1. Llama a la rutina `BARadialDistP2Matrices` para traducir el vector de parámetros a sus elementos $\mathbf{P} \rightarrow (\mathtt{P}^i, \boldsymbol{\kappa}^i, \ldots, \mathbf{X}_{aj})$
2. Calcula las proyecciones de las coordenadas afines 3D $\mathbf{X}_{aj}$ en las imágenes, según el modelo de proyección lineal habitual.
3. Distorsiona las coordenadas afines de los puntos proyectados según el modelo de distorsión radial, resultando $\hat{\mathbf{x}}_{adj}^i$. Esto lo hace la rutina `DistorsionRadial`.
4. Construir el vector de medidas $\widehat{\mathbf{X}}$, con las coordenadas afines distorsionadas de los puntos.

Optimizaciones. Según el valor de la variable `tipoOpt` se selecciona uno de los siguientes algoritmos.

`tipoOpt`	Tipo de Optimización
0	`fminunc`
	Levenberg-Marquardt
1	`LMB`
2	`LMSI`
3	`LMPB`
4	`LMPSI`
5	`LMPSI_BA`
6	`lsqnonlin`
7	`fminsearch`

Las opciones recomendadas son `LMSI` y `LMPSI_BA`. La optimización mediante `lsqnonlin` se realiza sobre la función `funBARadialDistReproy`, que es la que devuelve el vector `FUN`$(\mathbf{P}) = \mathbf{X} - f(\mathbf{P})$.

También es posible utilizar el comando `fminsearch`, que implementa el método del Simplex Cuesta Abajo (§ C.3.1), interno de MATLAB. Sólo necesita conocer la función de coste, y es bastante lento, al igual que `fminunc`, si hay muchas cámaras y puntos.

A pesar de que la expresión analítica de la matriz jacobiana es conocida, no se ha programado una rutina LM que incluya las derivadas exactas, principalmente porque las derivadas son complicadas y había cosas más interesantes que programar, ya que en la práctica la distorsión radial no es muy severa, sino que el modelo de proyección lineal es bastante bueno.

Todos las opciones antes presentadas calculan la matriz jacobiana de forma numérica y esto puede requerir un número considerable de evaluaciones de la función, dependiendo del tamaño del vector de parámetros **P** (número de cámaras y número de puntos), como se explica en D.1.4.

D.12. Autocalibración

D.12.1. Cámaras ortogonales

`CalibOrtCam`

Implementa el algoritmo de cámaras ortogonales de § 7.3. Produce solución exacta de autocalibración si las cámaras son ortogonales, aunque lo habitual es que se utilice como inicialización de la mayoría de algoritmos de autocalibración no lineales. Esta rutina es del proyecto ADREP-3D.

ENTRADA:

`P(3,4,ncam)`	matrices de proyección de la calibración proyectiva

SALIDA:

`C(4,4)`	cuádrica absoluta dual (en las coordenadas de la calibración proyectiva dada)
`pinf`	coordenadas del plano del infinito
`K(3,3,ncam)`	matrices de parámetros intrínsecos

D.12.2. Ecuaciones Kruppa

`CalibKruppa_01`

Implementa el algoritmo de autocalibración explicado en § 7.4. Es una modificación de una rutina previa del proyecto ADREP-3D. Se ha adaptado a la formulación en el marco común del algoritmo LM, según una función modelo, `modelo_kruppa`. La función de coste es `coste_kruppa`. Los datos auxiliares de la optimización son la descomposición en valores singulares de la matriz fundamental y se pasan a la función modelo mediante variables globales.

ENTRADA:

`P(3,4,ncam)`	matrices de proyección de la calibración proyectiva
`tipoOpt`	(opcional) tipo de optimización
`K0(3,3)`	(opcional) estimación inicial de la matriz de parámetros intrínsecos

SALIDA:

`K(3,3)`	matriz de parámetros intrínsecos
`coste`	coste algebraico mínimo alcanzado

Si no se proporciona un valor inicial de la matriz de parámetros intrínsecos, se toma el que devuelve el algoritmo de autocalibración de cámaras ortogonales.

Dado que la entrada son las matrices de proyección de una calibración proyectiva, el algoritmo calcula

todas las matrices fundamentales distintas posibles entre parejas de cámaras, calcula el coste algebraico y lo acumula. Si hay m cámaras, el número de parejas distintas es $\binom{m}{2} = m(m-1)/2$.

Optimizaciones. La rutina admite varios algoritmos de optimización, según el valor de `tipoOpt`. Por defecto está seleccionada la primera, `tipoOpt`=0.

`tipoOpt`	Tipo de Optimización
0	`fminunc`
	Levenberg-Marquardt
1	`LMB`
2	`LMSI`
3	`fminsearch`

D.12.3. Restricción unimodular

Se han programado dos versiones del algoritmo: la simplificada y la completa, ambas basadas la misma propiedad del módulo de los autovalores de la homografía del infinito.

`Calib_ModulusConstr_all`

Es la programación del algoritmo de autocalibración explicado en § 7.5, que incluye 6 términos de coste por cada cámara en el vector de medidas.

ENTRADA:

`P(3,4,ncam)`	matrices de proyección de la calibración proyectiva
`tipoOpt`	(opcional) tipo de optimización
`plano_inf0`	(opcional) estimación inicial del plano del infinito

SALIDA:

`plano_inf`	coordenadas del plano del infinito
`coste`	suma de cuadrados de coste en cada cámara

Si no se proporciona un valor inicial del plano del infinito, se toma el que resuelve el algoritmo de autocalibración de cámaras ortogonales.

Utiliza las funciones modelo `Modelo_ModulusConstr_all` y de coste `coste_ModulusConstr_all`. Antes de llamar a cualquiera de estas funciones según el esquema de optimización seleccionado, se hace un cambio de referencia proyectivo para que la primera matriz de proyección $\mathtt{P}^1$ sea la canónica en caso de no serlo previamente. Después de la optimización se deshace el cambio, aplicándolo a las coordenadas del plano del infinito estimado. El vector de parámetros objetivo es el nulo.

`tipoOpt`	Tipo de Optimización
0	`fminunc`
	Levenberg-Marquardt
1	`LMB`
2	`LMSI`
3	`lsqnonlin`
4	`LMBchi2`
5	`fminsearch`

`Calib_ModulusConstr`

Es la programación del algoritmo de autocalibración explicado en § 7.5, pero sólo incluyendo 2 términos de coste por cada cámara en el vector de medidas. En la situación esperada, el algoritmo ordena los autovalores de la homografía del infinito y halla las relaciones de módulo del autovalor real con los

autovalores complejos conjugados.

Es muy parecido al algoritmo `Calib_ModulusConstr_all`, salvo la anterior modificación. La nueva función modelo es `Modelo_ModulusConstr`, que acompaña a la función de coste `coste_ModulusConstr`.

D.12.4. Cuádrica Absoluta Dual

`QuasiLinear_Qinf`

Implementa el algoritmo explicado en § 7.6.1, que es el método cuasi-lineal de Bill Triggs para autocalibrar basándose en las restricciones de la proyección de la cuádrica absoluta dual. Todas las cámaras con los mismos parámetros intrínsecos.

ENTRADA:

`P(3,4,ncam>=4)`	matrices de proyección de la calibración proyectiva

SALIDA:

`Qinf(4,4)`	cuádrica absoluta dual (en las coordenadas de la calibración proyectiva dada)
`DIAC(3,3)`	matriz de la DIAC
`cost`	3 costes de los distintos pasos que da el algoritmo

Internamente llama a `MatrizDisenno_QwT` para calcular la matriz de diseño `A` a partir de las matrices de proyección. Esta rutina verifica que el número de cámaras es suficiente (≥ 4) antes de construir esa matriz, de dimensiones $15m \times 60$ y dimensiones mínimas 60×60.

`CalibTriggs`

Implementa el algoritmo no lineal explicado en § 7.6.1, para autocalibrar cámaras con los mismos parámetros intrínsecos utilizando las ecuaciones de la proyección de la cuádrica absoluta dual.

ENTRADA:

`P(3,4,ncam)`	matrices de proyección de la calibración proyectiva
`Qinf(4,4)`	(opcional) estimación inicial de la cuádrica absoluta dual
`DIAC(3,3)`	(opcional) estimación inicial de la matriz de la DIAC

SALIDA:

`Qinf(4,4)`	cuádrica absoluta dual
`DIAC(3,3)`	matriz de la DIAC
`K(3,3)`	matriz de parámetros intrínsecos
`cost`	costes de los distintos pasos que da el algoritmo

Si no se proporcionan alguno de los argumentos de entrada opcionales, se utilizan los que devuelve el algoritmo cuasi-lineal. Aunque el algoritmo pueda trabajar con 3 cámaras, a efectos prácticos se exigen 4 como mínimo.

La función `CosteAlgebraico_Kfija_wQ` calcula el coste como la norma del vector de error que devuelve la función modelo `ModeloAlgebraico_Kfija_wQ`. La función que devuelve el vector de restricciones que utiliza la rutina de programación dinámica secuencial `SQP` es `frest_Qinfw` y la que devuelve sus derivadas exactas es `frest_Grad_Qinfw`.

`Cross_Rn`

Se utiliza para calcular el "producto vectorial" de los vectores ω^* y $\mathtt{P}^i\mathtt{Q}^*_\infty\mathtt{P}^{i\top}$, resultado contenido en $\mathbb{R}^{15}$. Sirve como generalización del producto vectorial: dados dos vectores $\mathbf{w}$ y $\mathbf{v}$ de $\mathbb{R}^n$, calcula el

vector $\mathbf{u}$ de $\mathbb{R}^m$, $m = \binom{n}{2}$, mediante las fórmulas

$$u_{ij} = w_i v_j - w_j v_i \quad i = 1 \ldots n-1, \quad j = i+1 \ldots n$$

`CalibTriggsHartley`

Implementa el algoritmo no lineal explicado en § 7.6.2, para autocalibrar cámaras con los mismos parámetros intrínsecos utilizando las ecuaciones de la proyección de la cuádrica absoluta dual y una parametrización mediante el plano del infinito y la matriz de parámetros intrínsecos.

ENTRADA:

`P(3,4,ncam)`	matrices de proyección de la calibración proyectiva
`K0(3,3)`	(opcional) estimación inicial de la matriz de parámetros intrínsecos
`plano_inf0`	(opcional) estimación inicial del plano del infinito
`tipoOpt`	(opcional) tipo de optimización
`tipoFcoste`	(opcional) tipo de función de coste

SALIDA:

`K(3,3)`	matriz de parámetros intrínsecos
`plano_inf`	plano del infinito
`coste`	coste algebraico mínimo alcanzado
`H1(4,4)`	matriz de la homografía tal que $\mathtt{P}^1\mathtt{H1} = [\mathtt{I} \mid \mathbf{0}]$

Si no se proporcionan alguno de los argumentos de entrada opcionales, se utilizan los que devuelve el algoritmo cuasi-lineal. Aunque el algoritmo pueda trabajar con 3 cámaras, a efectos prácticos se exigen 4 como mínimo.

`DIAC_TriggsAutocal_Kfija`

Encapsula los algoritmos explicados en § 7.6, que estiman la cuádrica absoluta dual $\mathtt{Q}^*_\infty$ y la DIAC ω^* para cámaras con los mismos parámetros intrínsecos.

ENTRADA:

`P(3,4,ncam>=4)`	matrices de proyección de la calibración proyectiva
`K0(3,3)`	(opcional) estimación inicial de la matriz de parámetros intrínsecos
`plano_inf0`	(opcional) estimación inicial del plano del infinito
`q(5,1)`	(opcional) vector binario que indica los parámetros fijos de K0
`tipoParam`	tipo de parametrización de la optimización
`tipoOpt`	tipo de optimización

SALIDA:

`K(3,3)`	matriz de parámetros intrínsecos
`plano_inf`	plano del infinito
`coste`	coste algebraico mínimo alcanzado
`H1(4,4)`	matriz de la homografía tal que $\mathtt{P}^1\mathtt{H1} = [\mathtt{I} \mid \mathbf{0}]$

Las posiciones del vector `q(5,1)` hacen referencia a los parámetros intrínsecos $\mathrm{q} = [\alpha_v; u_0; v_0; \tau; \theta]$. Se utiliza para imponer restricciones durante la estimación, evitando que varíen algunos parámetros. Por ejemplo, $\mathtt{q} = [11100]$ es la restricción de píxeles de forma conocida ("cuadrados"), $\mathtt{q} = [10011]$ es la restricción de punto principal conocido.

Si `tipoParam`=0, entonces se utiliza el enfoque de Trigss, en el que el vector de parámetros es $\mathbf{P} = (\mathbf{q}^{*\top}, \boldsymbol{\omega}^\top)^\top$ y además se utiliza la SQP para imponer las restricciones. En cambio, si `tipoParam`=1, se busca la solución variando el plano del infinito y la matriz de parámetros intrínsecos.

D.12.5. Algoritmo de Hartley

`CalibHartley`

Implementa el algoritmo explicado en § 7.7, que realiza la estimación simultanea del plano del infinito y la matriz de parámetros intrínsecos, para cámaras con mismos parámetros intrínsecos.

ENTRADA:

`P(3,4,ncam>=4)`	matrices de proyección de la calibración proyectiva
`tipoOpt`	tipo de optimización
`plano_inf0`	(opcional) estimación inicial del plano del infinito
`K0(3,3)`	(opcional) estimación inicial de la matriz de parámetros intrínsecos

SALIDA:

`K(3,3)`	matriz de parámetros intrínsecos
`plano_inf`	plano del infinito
`coste`	coste algebraico mínimo alcanzado
`H1(4,4)`	matriz de la homografía tal que $\mathtt{P}^1\mathtt{H1} = [\mathtt{I} \mid \mathbf{0}]$
`R(3,3,ncam)`	matrices de rotación, en la referencia en que $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$

Al igual que en rutinas anteriores, si no hay algún dato del punto inicial de la búsqueda (valores del plano del infinito o la matriz de parámetros intrínsecos), se utiliza el dato correspondiente obtenido con el algoritmo cuasi-lineal de Triggs.

La función modelo se llama `ModeloIdentidad_Kfija` y calcula el vector de error dada una calibración proyectiva de las cámaras, junto con un vector de parámetros: plano π y matriz `K`.

$$f : (\pi, \mathtt{K}) \equiv \mathbf{P} \longrightarrow \epsilon = \left[\epsilon_1^\top \ \cdots \ \epsilon_i^\top \ \cdots \ \epsilon_m^\top\right]^\top$$

En concreto se ha elegido que ϵ_i sea la resta unitaria de las matrices X_i e Identidad, en forma de vector columna de 9 elementos. El vector de medidas es de tamaño $N = 9m$. La función de coste es `CosteIdentidad_Kfija`.

Se pueden aplicar varios algoritmos de optimización, pues las restricciones propias del problema han sido impuestas por una adecuada parametrización. Entre las distintas posibilidades están las rutinas `fminunc`, `LMB`, `LMSI`, `lsqnonlin` y `fminsearch`.

D.12.6. Horópteras

`calc_horop`

Calcula la matriz de la horóptera `H` asociada a un par de proyecciones, según la relación (7.21).

ENTRADA: las matrices de proyección `P1` y `P2` (3×4).

SALIDA: la matriz de la horóptera `Ho` de 4×4.

`Calib_ModulusConstr_Ho_all`

Implementa el algoritmo de autocalibración descrito en § 7.8.4. Se han programado dos versiones del algoritmo de la restricción unimodular: la simplificada y la completa. Esta rutina es el algoritmo completo.

ENTRADA:

`P(3,4,ncam)`	matrices de proyección
`tipoOpt`	(opcional) tipo de optimización
`tipoParam`	(opcional) tipo de parametrización del plano del infinito
`plano_inf0(4,1)`	(opcional) estimación inicial del plano del infinito

SALIDA:

`plano_inf(4,1)`	plano del infinito
`coste`	coste dado por la ec. (7.22)

La variable `tipoOpt` permite seleccionar una de las distintas rutinas de optimización, que son: `fminunc`, `LMB`, `LMSI`, `lsqnonlin`, `LMBChi2` y `fminsearch`.

Hay varias rutinas que calculan el coste, dependiendo de la parametrización elegida para el plano candidato, todas empiezan por `coste_unimodular_plano_all` y como datos auxiliares necesitan el plano y las matrices de las horópteras con las que intersecar. Las funciones modelo se llaman `Modelo_unimodular_plano_all` y son las que realmente ejecutan los cálculos. Devuelven vectores de error $\widehat{\mathbf{X}}$.

El tercer argumento de entrada, `tipoParam`, permite elegir una de las tres parametrizaciones del plano de prueba dadas en § 7.8.4.1 durante la optimización. Las rutinas que permiten las conversiones entre las tres parametrizaciones del plano de prueba son las siguientes:

- Las rutinas `Spherical2Cartesian`, `Cartesian2Spherical` y `normaliza_spherical` sirven para convertir entre coordenadas homogéneas $\mathbf{u}$ y coordenadas esféricas $\boldsymbol{\varphi}$. Estas tres rutinas son de propósito general, aunque se apliquen a este problema.
- Mientras que las rutinas `Plano2ParamCubicas` y `ParamCubicas2Plano` convierten coordenadas homogéneas $\mathbf{u}$ en instantes de corte reales con las horópteras $\mathbf{t}$ y viceversa, respectivamente.

`Calib_ModulusConstr_Ho`

Implementa el algoritmo de autocalibración simplificado. Posee la misma entrada-salida que el algoritmo completo, `Calib_ModulusConstr_Ho_all`, pero el coste es el de la ec. (7.25).

Las funciones modelo y de coste cambian, se les quita el sufijo `_all`. Empiezan, respectivamente, por `Modelo_unimodular_plano` y `coste_unimodular_plano`.

`Calib_Cinf_Ho`

Rutina principal de autocalibración basada en el algoritmo de estimación de la IAC explicado en § 7.8.5. Su entrada-salida es la siguiente:

ENTRADA:

`P(3,4,ncam)`	matrices de proyección
`tipoOpt`	(opcional) tipo de optimización
`tipoParam`	(opcional) tipo de parametrización del plano del infinito
`plano_inf0(4,1)`	(opcional) estimación inicial del plano del infinito

SALIDA:

`IAC(3,3)`	Proyección de la Cónica Absoluta
`plano_inf(4,1)`	plano del infinito, en coords. homogéneas
`coste`	la suma al cuadrado de costes dados por la ec (7.27), para cada proyección

El segundo parámetro de entrada, `tipoOpt`, permite elegir el tipo de optimización y con ello decidir si se desea minimizar el coste a través de la función de coste o a través de la función modelo mediante al algoritmo de Levenberg-Marquardt. La función de coste se puede optimizar con las rutinas `fminunc`, `fminsearch` y `LMBschi2`. La opción recomendada es utilizar `fminunc` (`tipoOpt` = 0). La utilización

de la función modelo no supone ventaja alguna.

El tercer argumento de entrada, `tipoParam`, permite elegir una de las tres parametrizaciones del plano de prueba dadas en § 7.8.4.1 durante la optimización.

El cuarto parámetro de entrada, `plano_inf0`, proporciona una inicialización de la búsqueda no lineal del plano del infinito, parametrizado en coordenadas homogéneas. En caso de no proporcionar vector de parámetros inicial de la búsqueda, se utiliza el plano del infinito calculado bajo la suposición de cámaras "ortogonales" (`CalibOrtCam`): punto principal en el origen de coordenadas y píxeles cuadrados.

Las rutinas `modelo_Conica_plano` y `coste_Conica_plano` implementan las funciones modelo y de coste para la parametrización de coordenadas homogéneas, respectivamente. A su vez, llaman a otras rutinas: `model_ajuste_conica` y `cost_ajuste_conica`, respectivamente. Por último, ambas llaman a `MatrizDisennoConica` para construir la matriz de diseño del enfoque SVD con $\mathtt{A} = \mathtt{A}_{\mathbb{R}}$.

`Calib_Qinf_Ho`

Todo el desarrollo explicado en § 7.8.6 se puede programar en pocas líneas de código. La rutina principal de calibración del este algoritmo es `Calib_Qinf_Ho`. La entrada es la misma que la del algoritmo de ajuste de la cónica.

ENTRADA:

`P(3,4,ncam)`	matrices de proyección
`tipoOpt`	(opcional) tipo de optimización
`tipoParam`	(opcional) tipo de parametrización del plano del infinito
`plano_inf0(4,1)`	(opcional) estimación inicial del plano del infinito

SALIDA:

`Qinf(4,4)`	Cuádrica Absoluta Dual
`plano_inf(4,1)`	plano del infinito, en coords. homogéneas
`coste`	coste algebraico del ajuste, es decir, menor valor singular

Las rutinas `modelo_Cuadrica_plano` y `coste_Cuadrica_plano` implementan las funciones modelo y de coste para la parametrización de coordenadas homogéneas, respectivamente. A su vez, llaman a otras rutinas: `model_ajuste_cuad` y `cost_ajuste_cuad`, respectivamente. Por último, ambas llaman a `Ptos2HazPlanos` para calcular dos planos base del haz por cada pareja de punto de corte real y complejo, junto con la rutina `MatrizDisennoCuadrica` para construir la matriz de diseño del enfoque SVD con $\mathtt{A} = \mathtt{A}_{\mathbb{R}}$.

La salida incluye $\mathtt{Q}^*_\infty$, propio de este algoritmo de estimación, junto con su núcleo π_∞ y su coste asociado. El ajuste de la cuádrica es un algoritmo lineal (DLT o SVD) con restricciones, en concreto, el enfoque SVD encaja con el algoritmo A3.6, pág 565 de [1], programado en la rutina `ConstrainedMin1`.

D.13. Ajuste de haces euclídeo

`BundleAdjustmentEuclideo`

Optimiza una reconstrucción euclídea minimizando el error de reproyección, según se describe en § 8.2. Realiza un encapsulado similar a `BundleAdjustment`: a partir de los datos, ofrece distintos caminos de optimización, según el valor de la variable de entrada `tipoOpt`. Los otros argumentos de entrada son los puntos observados en las imágenes, las matrices de proyección tras la homografía resultado de la autocalibración y los puntos 3D asociados.

ENTRADA:

`P`	matrices de proyección 'euclídeas'
`X3d`	coordenadas afines de los puntos 3D 'euclídeos'
`x`	coordenadas afines de los puntos observados
`tipoOpt`	tipo de optimización

SALIDA:

`X3dmin`	coordenadas afines de los puntos 3D euclídeos
`pmin`	matrices de proyección euclídeas
`xc`	coordenadas afines de los puntos proyectados corregidos
`coste`	distancia geométrica mínima alcanzada, medida en píxeles

La función de coste `Coste_BAEuclideo` calcula el error de reproyección de los npuntos puntos en las ncam imágenes llamando a la función modelo, `BAEuclideo_ErrorReproy`, que a su vez llama a la rutina `BAEuclidP2Matrices` para traducir el vector de parámetros a matrices de proyección y puntos 3D, $\mathbf{P} \rightarrow (\mathtt{P}^i_{\mathrm{M}}, \ldots, \mathbf{X}_{\mathrm{M}\,aj})$. Una vez que construye las matrices de proyección a partir de los parámetros euclídeos de las mismas, aplica el modelo de proyección lineal.

La ventaja de esta rutina es que permite emplear distintas parametrizaciones de las matrices de proyección, no sólo la descrita en § 8.2.1. Basta con especificar dentro de la rutina:

1. El nombre de la función que realiza el paso inverso: dada una matriz de proyección euclídea, obtiene sus parámetros $\mathbf{a}^j$. Necesaria para inicializar la búsqueda del mínimo en un punto cercano.
2. El nombre de la función que transforma el vector de parámetros euclídeos $\mathbf{a}^j$ de cada cámara en la matriz de proyección. Necesaria para traducir el vector de parámetros del mínimo a las matrices de proyección.
3. El tamaño del vector de parámetros de cada cámara, $\mathbf{a}^j$.

Las funciones que permiten la paremetrización y desparametrización actual son `ParametrosEuclid2P_2` para pasar de los parámetros euclídeos a las matrices de proyección, y `P2ParametrosEuclid_2`, para realiza el paso inverso. El tamaño del vector de parámetros de cada cámara es 9. Aunque la parametrización actual funciona, faltaría probar otras y comparar resultados y rendimiento.

Optimizaciones. Siendo conscientes de la existencia de múltiples parametrizaciones de las matrices de proyección, se decidió implementar una versión del algoritmo LM que fuera aplicable a cualquier ajuste de haces. El resultado fue la rutina `LMPSI_BA`, que como es un caso general, estima la matriz jacobiana numéricamente.

Además, la rutina `LMSI` es aplicable a cualquier problema disperso y utiliza las funciones propias de MATLAB del *sparse toolbox*. Esta también estima numéricamente la matriz jacobiana y luego la convierte a una estructura de matriz dispersa de MATLAB. El propio MATLAB se encarga de realizar las operaciones, manteniendo la estructura dispersa mientras pueda.

Según el valor de la variable `tipoOpt` se selecciona uno de los siguientes algoritmos.

`tipoOpt`	Tipo de Optimización
0	`fminunc`
	Levenberg-Marquardt
1	`LMB`
2	`LMSI`
3	`LMPB`
4	`LMPSI`
5	`LMPSI_BA`
6	`LMPSBAI_E`
7	`lsqnonlin`
8	`fminsearch`

La opción recomendada es `LMPSBAI_E` porque calcula la matriz jacobiana de forma exacta. Si se quiere evaluar de forma numérica la matriz jacobiana, las opciones recomendadas son `LMSI` y `LMPSI_BA`.

La optimización mediante `lsqnonlin` se realiza sobre la función `funBAEuclideoReproy`, que es la que devuelve el vector $\texttt{FUN}(\mathbf{P}) = \mathbf{X} - f(\mathbf{P})$. También se ha incluido la posibilidad de utilizar el comando `fminsearch`, que implementa el método del Símplex Cuesta Abajo (§ C.3.1), interno de MATLAB, mas no es una buena idea utilizarlo.

Las derivadas exactas se calculan en varios pasos. La función `Jacob_MProjEuclidea` calcula la matriz jacobiana $\texttt{J}_g^i$ de una cámara euclídea y para construir las derivadas respecto de los parámetros de la rotación llama a la rutina `DerivadasRotacion`. Ésta calcula la matriz 9×3 de derivadas parciales `dR` (de cada elemento de una matriz de rotación `R`, respecto de cada elemento del vector $\mathbf{u}$ que la parametriza, según la matriz exponencial).

D.14. Rutinas Varias

Ajustar una cónica a unos puntos del plano

`ajustar_conica`

Es una función que implementa un algoritmo lineal (DLT o SVD) para ajustar una cónica a unos puntos proyectivos dados.

Fundamentos del algoritmo
Siguiendo con el algoritmo de § 3.2.5, cinco puntos definen una cónica C unívocamente. Si hay $n > 5$ puntos nos planteamos el siguiente problema de optimización : encontrar la cónica C tal que

$$\min_C \sum_i^n |\mathbf{p}_i^\top C \mathbf{p}_i|^2$$

Siguiendo la misma idea que entonces: se ponen las condiciones de cada punto una a continuación de otra y así se forma la matriz de diseño `A`, de $n \times 6$, el problema se formula de la siguiente manera

$$\min_C \sum_i^n |\mathbf{p}_i^\top C \mathbf{p}_i|^2 = \min_{\mathbf{c}} \sum_i^n |\mathbf{q}_i^\top \mathbf{c}|^2 \equiv \min_{\mathbf{c}} \|\mathtt{A}\mathbf{c}\| \quad \text{sujeto a} \quad \|\mathbf{c}\| = 1$$

El sistema homogéneo asociado es: $\mathtt{A}\mathbf{c} = \mathbf{0}$, siendo

$$\mathtt{A} = \begin{bmatrix} \mathbf{q}_1^\top \\ \vdots \\ \mathbf{q}_n^\top \end{bmatrix}$$

El problema y su solución son clásicos, similares a los comentados en el caso del algoritmo de los ocho puntos para la matriz fundamental. La solución se obtiene mediante la SVD de $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^\top$. La cónica $\mathbf{c}$ es la columna de `V` correspondiente al menor valor singular de `A`, es decir, la sexta columna. Si los puntos son complejos y se proporcionan conjugados 2 a 2, la matriz C es real.

`AjusteCuadrica`

Es una rutina del proyecto ADREP-3D que realiza un ajuste similar, pero incluye más posibilidades: ajuste en $\mathbb{P}^n$ y la búsqueda del mínimo en un subespacio del ambiente. En el proyecto se ha modificado la rutina para incluir condiciones de conjugación respecto de la cuádrica (ortogonalidad de rectas si se desea aplicar a la estimación de la matriz Σ del Calibration Pencil, § 3.4.5).

Cálculo del núcleo de una matriz

La rutina `NucleoDcha` calcula el núcleo por la derecha de una matriz `A`, es decir, el vector $\mathbf{e}$ tal que $\mathtt{A}\mathbf{e} = \mathbf{0}$ (si es posible). Es igual que la rutina `NumKernel` de ADREP-3D, la cual devuelve un vector unitario asociado al menor valor singular de la matriz A y el menor valor singular como medida del ajuste. La única diferencia es que `NucleoDcha` sólo devuelve el vector, no el valor singular.

La rutina es muy útil, pues sirve para hallar epipolos si `A` es una matriz Fundamental `F`. El epipolo en la primera imagen verifica $\mathtt{F}\mathbf{e}_1 = \mathbf{0}$ y el epipolo en la segunda imagen, $\mathtt{F}^\top\mathbf{e}_2 = \mathbf{0}$. También sirve para calcular el centro óptico de una cámara dada una matriz de proyección `P` 3×4, ya que el centro óptico es el vector que verifica $\mathtt{P}\mathbf{C} = \mathbf{0}$.

Normalización de las coordenadas homogéneas

`NormaUnidadxCol`

ENTRADA: matriz P, cuyas columnas son las coordenadas homogéneas de puntos en $\mathbb{P}^n$.
REALIZA: una normalización de las columnas de P a norma unidad. Matricialmente, es muy fácil de programar en MATLAB mediante `repmat`.
SALIDA: las coordenadas homogéneas normalizadas.

Es equivalente a la rutina `UnitarizaCoords` del proyecto ADREP-3D. Sirve para normalizar las coordenadas homogéneas de los puntos de $\mathbb{P}^n$, hacerlas de módulo unidad.

`randmuestra`

Genera un vector de m números naturales aleatorios distintos, seleccionados entre los naturales del intervalo $1 \ldots N$; para ello llama a la rutina `randperm` de MATLAB y se queda con m elementos.

`IsVectInMatrix`

Comprueba si un vector fila está en una de las filas de una matriz, sin importar el orden de los elementos del vector. Sirve para llevar la cuenta de las muestras de puntos probadas en los algoritmos RANSAC, y así no repetir ninguna.

Momentos de una distribución de puntos 3D

`CovarianceMoments`

Calcula la matriz de varianzas-covarianzas de una distribución uniforme Q de elementos de 3 componentes:

$$\sum_i \begin{bmatrix} x_i^2 & x_i y_i & x_i z_i \\ y_i x_i & y_i^2 & y_i z_i \\ z_i x_i & z_i y_i & z_i^2 \end{bmatrix} \tag{D.7}$$

Calcula la SVD de (D.7) para diagonalizarla, obtener los "momentos principales" y el cambio de base de los puntos, $Q' = \mathtt{R}^{-1}Q$ tal que tal que los nuevos "momentos principales" estén alineados con los ejes coordenados. Esta rutina se utiliza para facilitar la comparación del error de reconstrucción 3D entre la distribución de puntos 3D original y la reconstruida.

`PrincipalMoments`

Realiza lo mismo pero respecto del tensor de inercia:

$$\sum_i \begin{bmatrix} y_i^2+z_i^2 & -x_iy_i & -x_iz_i \\ -y_ix_i & z_i^2+x_i^2 & -y_iz_i \\ -z_ix_i & -z_iy_i & x_i^2+y_i^2 \end{bmatrix} \tag{D.8}$$

Este tensor no sirve para transformaciones de semejanza que incluyan escalado. Recordemos de § 3.3.3 que las transformaciones que sólo incluyen rotación y traslación reciben el apellido de Euclídeas. Es decir, el algoritmo no sirve para comparar distribuciones de puntos que tengan distintas escalas (transformaciones de semejanza).

Obtención de la matriz de parámetros intrínsecos

`CholeskyKKT`

Realiza la descomposición de Cholesky mediante la SVD y la descomposición RQ. Su función consiste en descomponer una matriz `A` de 3×3 simétrica y definida positiva según la fórmula $\mathtt{A} = \mathtt{K}\mathtt{K}^\top = (\mathtt{K}\mathtt{Q}) * (\mathtt{Q}^\top\mathtt{K}^\top)$, siendo `K` una matriz triangular superior y Q una matriz ortogonal. Más detalles en [1, pág. 556]. Resultado A3.5. "Positive-definite symmetric real matrix".

Por precaución, se hace la descomposición SVD sobre la parte simétrica de $\mathrm{Re}\,(\mathtt{A})$, que todavía puede no ser definida positiva, mas la rutina devuelve resultado aunque no se cumpla.

`DIAC2K`

Obtiene la matriz de parámetros intrínsecos a partir de la DIAC. El método empleado es realizar la descomposición de Cholesky mediante SVD, seguida de la descomposición RQ, es decir, llamar a `CholeskyKKT`, como se indica en [1].

`IAC2K`

Obtiene la matriz de parámetros intrínsecos a partir de la IAC o PAC. El método empleado consiste en emplear la descomposición de Cholesky y/o la descomposición en autovalores y autovectores de MATLAB. Existen dos alternativas. Además, la rutina devuelve para cada matriz pasada (como candidata a "IAC") un flag que indica si es definida positiva (1), no definida (0), o definida negativa (-1).

Cambios de referencia

`AplicaCambio`

Aplica una homografía del espacio a las matrices de proyección y puntos 3D de una calibración proyectiva, conservando los puntos proyectados. En un cambio de referencia dado por la homografía de matriz `C`, se verifica

$$\lambda\mathbf{x} = \mathtt{P}\mathbf{X} = (\mathtt{P}\mathtt{C})(\mathtt{C}^{-1}\mathbf{X}) = \tilde{\mathtt{P}}\tilde{\mathbf{X}}$$

Si los puntos 3D se transforman de acuerdo a $\tilde{\mathbf{X}} = \mathtt{C}^{-1}\mathbf{X}$, las matrices de proyección lo hacen según $\tilde{\mathtt{P}} = \mathtt{P}\mathtt{C}$.

`AbsQuad2H`

Obtiene la matriz de la homografía del espacio que transforma una calibración proyectiva en una calibración métrica. Está basada en el resultado 18.1 de [1, pág. 444-447] y para ello la primera matriz de proyección debe ser la canónica $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$.

`GeneralChangeBaseMatrix`

Calcula la matriz de la aplicación proyectiva de dimensión n que lleva cada punto de la referencia canónica a las columnas de la matriz pasada como argumento. La rutina `ChangeBaseMatrix` de ADREP-3D realiza lo mismo, pero sólo para $\mathbb{P}^3$.

`SeleccionaBase`

Busca entre un conjunto de puntos dados, aquellos que no estén aproximadamente alineados ($\sim$ linealmente dependientes). Todavía no está muy depurada la condición de alineamiento en forma numérica.

Coordenadas de rectas de $\mathbb{P}^3$

La rutina `PermutaPlucker` sirve para pasar de coordenadas de Plücker de plano de una recta a coordenadas de Plücker de punto de la misma, a través de las matrices antisimétricas que las representan.

Optimización lineal con restricciones

`ConstrainedMin1`

Implementa el algoritmo A3.6 de [1, pág. 565], que resuelve el problema

$$\min_{\mathbf{x}} \|\mathtt{A}\mathbf{x}\| \quad \text{sujeto a} \quad \|\mathbf{x}\| = 1 \ \text{ y } \ \mathtt{C}\mathbf{x} = \mathbf{0}$$

`ConstrainedMin2`

Resuelve el problema

$$\min_{\mathbf{x}} \|\mathtt{A}\mathbf{x}\| \quad \text{sujeto a} \quad \|\mathbf{x}\| = 1 \ \text{ y } \ \mathbf{x} = \mathtt{E}\mathbf{a}$$

La solución está detallada en [1, pág. 566], algoritmo A3.7. La rutina acepta como tercer argumento de entrada, `rankE`, el rango teórico de la matriz de restricción `E`, en caso de no proporcionar este dato, se toma el rango del comando `rank` de MATLAB.

`ConstrainedMin3`

Implementa el algoritmo A3.8 de [1, pág. 567], que resuelve el problema

$$\min_{\mathbf{x}} \|\mathtt{A}\mathbf{x}\| \quad \text{sujeto a} \quad \|\mathtt{C}\mathbf{x}\| = 1$$

Programación cuadrática secuencial

`SQP`

Esta rutina (Sequential Quadratic Programming) implementa algoritmo de optimización general como puede ser `LMBchi2`, pero con los ingredientes propios del problema del que recibe el nombre. Las indicaciones sobre la implementación se obtuvieron de [15].

ENTRADA:

`P0`	Estimación inicial del vector de parámetros $\mathbf{P}$
`fcoste`	función de coste a minimizar $g(\mathbf{P})$
`arg_fcoste`	argumentos opcionales que necesite la función fcoste
`frest`	función vectorial de las restricciones $\mathbf{c}(\mathbf{P})$
`frest_grad`	función que devuelve la matriz jacobiana exacta de las restricciones $\mathbf{c}(\mathbf{P})$

En caso de no proporcionar un nombre para la rutina que devuelve la matriz jacobiana exacta, esta se calcula de forma numérica.

OBJETIVO: encontrar el vector de parámetros **P** que minimiza la función de coste y verifica las restricciones **c**, siguiendo el algoritmo de Newton.

SALIDA:

`POmin`	vector de parámetros del mínimo
`coste_min`	valor del mínimo (de la función de coste anterior)

`SQP_LM`

Es una variación sobre la rutina anterior, la cual incluye la modificación de Levenberg-Marquardt del Hessiano. Como todas las rutinas LM tiene un parámetro λ que controla las dos fases de exploración y explotación, mejorando el esquema de Newton. Posee la misma entrada-salida que la rutina `SQP`.

Índice alfabético

Bibliografía

[1] R. I. Hartley, A. Zisserman,
Multiple View Geometry in Computer Vision,
Cambrigde University Press, 2000.

[2] R. I. Hartley,
Computation of the Quadrifocal Tensor.
ECCV 1998: pp 20-35.

[3] R. I. Hartley,
Minimizing Algebraic Error in Geometric Estimation Problems.
ICCV 1998, pp. 469-476.

[4] Richard I. Hartley, Peter F. Sturm,
Triangulation.
CAIP 1995, pp. 190-197.

[5] R. I. Hartley,
In Defence of the 8-Point Algorithm.
ICCV 1995, pp. 1064-1070.

[6] R. I. Hartley,
Euclidean Reconstruction from uncalibrated views,
Applications of Invariance in Computer Vision 1993: 237-256.

[7] F. Schaffalitzky, A. Zisserman, R. I. Hartley, P. H. S. Torr,
A Six Point Solution for Structure and Motion.
ECCV 2000, pp. 632-648.

[8] J. I. Ronda,
Notas de Geometría para reconstrucción 3D,
Grupo de Tratamiento de Imágenes, ETSIT.

[9] J. I. Ronda, A. Valdés, F. Jaureguizar,
Camera autocalibration and horopter curves.
Documento interno proyecto ADREP-3D. Se puclicará en el IJCV.

[10] A. Valdés, J. I. Ronda,
Camera autocalibration and the calibration pencil.
Documento interno proyecto ADREP-3D.

[11] J.Ponce,
On Computing Metric Upgrades of Projective Reconstructions Under the Rectangular Pixel Assumption.
Proc. of the SMILE 2000 Workshop on 3D Structure from Multiple Images of Large-Scale Environments, Dublin, Ireland, Junio 2000. Springer-Verlag Notes in Computer Science 2018, M. Pollefeys, L.Van Gool, A. Zisserman y A. Fitzgibbon (Eds.), pp 52-67.

[12] Yi Ma, S. Soatto, J. Kosecká, S. S. Sastry,
An Invitation to 3-D Vision. From Images to Geometric Models.
Springer, 2004.

[13] J.G. Semple, G.T. Kneebone,
Algebraic Projective Geometry,
Oxford University Press, 1998.

[14] J. M. Rodríguez-Sanjurjo, J. M. Ruiz,
Geometría Proyectiva.
Addison-Wesley, 1998.

[15] B. Triggs,
Autocalibration and the Absolute Quadric,
Proc IEEE Conference on Computer Vision and Pattern Recognition, pp 609-614. CVPR 97.

[16] B. Triggs, P. F. McLauchlan, R. I. Hartley, A. W. Fitzgibbon,
Bundle Adjustment - A Modern Synthesis.
Workshop on Vision Algorithms 1999: 298-372.

[17] A. Heyden,
Lectures on Projective Geometry,
EEFN (First Frensh-Nordic Summerschool in Geometry), Junio 2001.

[18] A. Heyden,
Tensorial Properties of Multilinear Constraints,
Mathematical Methods in the Applied Sciences 23, pp. 169-202, 2000.

[19] A. Heyden,
Tutorial on Multiple View Geometry,
In cunjunction with ICPR'00, Sept. 2000.

[20] A. Heyden,
Tutorial on Tensorial Description of Multiple View Geometry
In cunjunction with ICCV'99, Sept. 1999.

[21] A. Heyden,
A Common Framework for Multiple-View Tensors,
European Conference on Computer Vision, Freiburg, Germany, 1998.

[22] A. Heyden,
Reduced Multilinear Constraints - Theory and Experiments,
International Journal of Computer Vision, 30 1 pp. 5-26, 1998.

[23] A. Heyden, K. Åström,
Euclidean Reconstruction from Constant Intrinsic Parameters,
International Conference on Pattern Recognition, Vienna, Austria, volumen 1, pp. 339-343, 1996.
IEEE Computer Society Press.

[24] A. Heyden, K. Åström,
A Canonical Framework for Sequences of Images,
IEEE Workshop on Representation of Visual Scenes, MIT, Boston, MA, pp. 45-52, 1995. IEEE Computer Society Press.

[25] K. Åström, A. Heyden, F. Kahl, R. Berthilsson, G. Sparr,
A Computer Vision Toolbox,
Nordic Matlab Conference, Stockholm, 1997.

[26] M. Pollefeys,
Self-calibration and metric 3D reconstruction from uncalibrated image sequences,
PhD, Mayo 1999. Katholieke Universiteit Leuven - Faculteit Toegepaste Wetenschappen Arenbergkasteel, B-3001 Heverlee (Belgium).

[27] M. Pollefeys, L. Van Gool, A. Oosterlinck,
The Modulus Constraint: A New Constraint for Self-Calibrarion.
Proceedings of ICPR '96.

[28] E. Trucco, A. Verri,
Introductory Techniques for 3-D Computer Vision,
Prentice Hall, primera edición, Marzo 1998.

[29] S. Birchfield,
An Introduction to Projective Geometry (for computer vision),
Stanford University, Marzo 1998.

[30] Zhengyou Zhang,
Determining the Epipolar Geometry and its Uncertainty: A Review,
INRIA Research Report No. 2927, Julio 1996.

[31] W. Press, B. Flannery, S. Teukolsky, W. Vetterling,
Numerical Recipes in C.
Cambridge University Press, 1988.

[32] H. S. M. Coxeter,
Projective Geometry.
Toronto: University of Toronto, segunda edición, 1974.

[33] O. Faugeras,
Three-Dimensional Computer Vision.
Cambridge, MA: MIT Press, 1993.

[34] L. Guibas,
Lecture notes for CS348a: Computer Graphics - Mathematical Foundations.
Stanford University, Otoño 1996.

[35] Q.-T. Luong and O. D. Faugeras,
Fundamental matrix: Theory, algorithms, and stability analysis.
International Journal of Computer Vision, 17(1):43-75, 1996.

[36] J. L. Mundy and A. Zisserman,
Geometric Invariance in Computer Vision.
Cambridge, MA: MIT Press, 1992.

[37] Z. Zhang and G. Xu.,
Epipolar Geometry in Stereo, Motion and Object Recognition: A Unified Approach.
Kluwer Academic Publishers, 1996.

[38] Ben Noble and James W. Daniel,
Applied Linear Algebra,
Prentice Hall, 3ł edición, Noviembre 1987.

[39] Leslie Lamport,
LaTeX: A Document Preparation System,
Addison-Wesley, 1986.

[40] M. Goossens, F. Mittelbach and A. Samarin,
The LaTeXCompanion
Addison Wesley, 2000.

[41] R. J. Valkenburg and P. L. Evans,
Lens Distorsion Calibration by Straightening Lines
Proceedings of Image and Vision Computing New Zealand 2002.

[42] Gergely Vass and Tamás Perlaki,
Applying and removing lens distorion in post production
Budapest University of Technology.

[43] W. Chojnacki, M. J. Brooks, A. van den Hengel y D. Gawley,
Revisiting HartleySs Normalised Eight-Point Algorithm.
IEEE Trans. Pattern Analysis Machine Intelligence, 25, 9, 2003, pp 1172-1177.

[44] A. van den Hengel, W. Chojnacki, M. J. Brooks, D. Gawley,
A new constrained parameter estimator: experiments in fundamental matrix computation.
BMVC 2002.

Pliego de Condiciones

Este documento contiene las condiciones legales que guiarán la implementación, en este proyecto, de los módulos que realicen la autocalibración y la reconstrucción 3D. En lo que sigue se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora, con la finalidad de llevar a cabo el desarrollo comentado. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto considerado. Esta línea de investigación, junto con el posterior desarrollo de los programas, está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

- Condiciones generales

  1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.
  2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.
  3. En la oferta se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si éste se hubiera fijado.
  4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.
  5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.
  6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.
  7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras, siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades, sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.
  8. Tanto en las certificaciones de obras como en la liquidación final se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si, excepcionalmente, se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata, pero que sin embargo fuera admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa, y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.
10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere, y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento se sujetarán siempre a lo establecido en el punto anterior.
11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio, o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, sólo tendrá derecho, sin embargo, a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.
12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.
13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras, así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.
14. Concluida la ejecución de la obra será reconocida por el Ingeniero Director que a tal efecto designe la empresa.
15. La garantía definitiva será del 4 % del presupuesto, y la provisional del 2 %.
16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.
17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas, y la definitiva, al año de haber ejecutado la provisional, procediéndose, si no existe reclamación alguna, a la reclamación de la fianza.
18. Si el contratista, al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.
19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo lo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto el contratista deberá consultarle cualquier duda que surja en su realización.
20. Durante la realización de la obra se girarán visitas de inspección por personal facultativo de la empresa cliente para hacer las comprobaciones que se crean oportunas. Es obligación del contratista la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.
21. El contratista deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa por retraso de la ejecución, siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.
23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata", anteriormente llamado "Presupuesto de Ejecución Material"que hoy designa otro concepto.

- Condiciones particulares La empresa consultora que ha desarrollado el presente proyecto lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

  1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.
  2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación, bien para su uso en trabajos o proyectos posteriores para la misma empresa cliente o para otra.
  3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.
  4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones, así como su cantidad.
  5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.
  6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él deberá ser notificada al Ingeniero Director del Proyecto, y a criterio de éste la empresa consultora decidirá aceptar o no la modificación propuesta.
  7. Si la modificación se acepta la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.
  8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.
  9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto deberá comunicarlo a la empresa consultora.
  10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.
  11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.
  12. El Ingeniero Director del presente proyecto será el responsable de la dirección de la aplicación industrial, siempre que la empresa consultora lo estime oportuno. En caso contrario la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.

Presupuesto

- Ejecución Material
  - Compra de un ordenador de sobremesa (software incluido) ..... 2.500 €
  - Compra de una cámara fotográfica digital (Sony Cyber-shot DSC-F717) ..... 850 €
  - Alquiler de una impresora láser durante 1 mes ..... 180 €
  - Material de oficina ..... 150 €
  - Subtotal de ejecución material ..... 3.680 €
- Gastos Generales
  - 16 % sobre Ejecución Material ..... 589 €
- Beneficio Industrial
  - 6 % sobre Ejecución Material ..... 221 €
- Gastos de personal
  - 8 h/día · 20 días/mes · 8 meses = 1280 horas
  - gasto (directo e indirecto) por hora de trabajo de ingeniero: 23 €/hora ..... 29.440 €
- Material Fungible
  - Gastos de impresión ..... 30 €
  - Encuadernación ..... 150 €
  - Subtotal de material fungible ..... 180 €
- Subtotal del presupuesto
  - Subtotal Presupuesto ..... 34.110 €
- I.V.A. aplicable
  - 16 % Subtotal Presupuesto ..... 5.458 €
- Total Presupuesto
  - Total Presupuesto ..... 39.568 €

Madrid, a 23 de febrero de 2004.

El Ingeniero Jefe del Proyecto

Fdo: Guillermo Gallego Bonet

Ingeniero de Telecomunicación